VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Lenka Babjarčiková

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH NOVÝCH LABORATORNÍCH ÚLOH PRO VÝUKU PRINCIPŮ KOMUNIKAČNÍCH PROTOKOLŮ

CREATION OF NEW LABORATORY EXERCISES EXPLAINING THE PRINCIPLES OF COMMUNICATION PROTOCOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Lenka Babjarčiková**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Lukáš Langhammer, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Bc. Lenka Babjarčiková ***ID:*** 185884
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Návrh nových laboratorních úloh pro výuku principů komunikačních protokolů

POKYNY PRO VYPRACOVÁNÍ:

Zadání práce spočívá v návrhu dvou nových laboratorních úloh v simulačním prostředí NS3. Prostudujte možnosti simulačního prostředí NS3 a na základě možností tohoto prostředí vytvořte kompletní návody vhodné pro studenty zahrnující výchozí scénáře, doplňující úkoly a kontrolní otázky. Časová náročnost každé úlohy musí být přibližně dvě hodiny. Při návrhu úloh se zaměřte na vysvětlení principů komunikačních protokolů (nabízí se například ARP, DHCP, NAT, DNS, BGP, protokoly sady IPv6 aj.).

DOPORUČENÁ LITERATURA:

[1] FOROUZAN, Behrouz A. TCP/IP protocolsuite. 4th ed. Boston: McGraw-HillHigherEducation, 2010, xxxv, 979 s. ISBN 978-0-07-337604-2.

[2] NS3 Consortium: Dokumentace k Network Simulator 3 (NS3), 2014. Dostupné online <http://www.nsnam.org/documentation/>, citováno 10.9.2020.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Lukáš Langhammer, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cieľom diplomovej práce bolo vytvoriť dve laboratórne úlohy pre vysvetlenie princípov vybraných komunikačných protokolov v simulačnom prostredí ns-3. Vybranými protokolmi boli BGP a ICMPv6. Prvá kapitola práce obsahuje teoretický opis fungovania vybraných protokolov a ich správ. Pre protokol BGP sú v tejto kapitole zhrnuté vlastnosti troch možností implementácie v prostredí ns-3. Druhú časť práce tvoria vytvorené laboratórne úlohy. Úloha pre protokol BGP opisuje spôsob akým si smerovače vytvárajú susedstvá na úrovni autonómnych systémov, ako v rámci týchto susedstiev komunikujú a ako protokol reaguje na výpadok prepojenia susedných smerovačov. Úloha pre protokol ICMPv6 sa zameriava na správy protokolu a ich využitie v rámci viacerých funkcionalít protokolu ako je napríklad konfigurácia adries, získanie linkovej adresy zaradenia či hlásenia chýb v sieti.

KĽÚČOVÉ SLOVÁ

AS, BGP, DCE, ICMPv6, IPv6, Quagga, ns-3

ABSTRACT

The aim of this master thesis is to create two laboratory exercises in ns-3 network simulator for chosen communication protocols. The chosen protocols are BGP and ICMPv6. The first part of this thesis contains teoretical knowledge used in order to design laboratory exercise. It contains descriptions of funcionality of both protocols and messages they use. This chapter also includes brief description of three options which were considered for implementation of BGP protocol into ns-3 enviroment. The second chapter consists of actual laboratory exercises which were created as a purpose of this thesis. Exercise for BGP protocol deals with creating neighborhoods between routers at autonomous system level, analyzing the way routers communicate within their neighborhoods and how protocol reacts when the connection between two neighbor routers fails. Exercise created for ICMPv6 protocol is focused on its messages and their usage within multiple protocol funcionalities, such as address configuration, link address resolution and error detection.

KEYWORDS

AS, BGP, DCE, ICMPv6, IPv6, Quagga, ns-3

BABJARČIKOVÁ, Lenka. *Návrh nových laboratorních úloh pro výuku principů komunikačních protokolů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 110 s. Diplomová práca. Vedúci práce: Ing. Lukáš Langhammer, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Bc. Lenka Babjarčiková
VUT ID autora:	185884
Typ práce:	Diplomová práca
Akademický rok:	2020/21
Téma závěrečnej práce:	Návrh nových laboratorních úloh pro výuku principů komunikačních protokolů

Vyhlasujem, že svoju záverečnú prácu som vypracovala samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autorka uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušila autorské práva tretích osôb, najmä som nezasiahla nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomá následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autorky*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rada by som sa chcela poďakovať svojim rodičom a priateľom, za ich podporu počas celého štúdia a vedúcemu práce pánu Ing. Lukášovi Langhammerovi, Ph.D. za jeho odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Zoznam obrázkov

Úvod

Cieľom tejto diplomovej práce bolo navrhnúť a vytvoriť dve laboratórne úlohy v simulačnom prostredí ns-3. Úlohy sa mali zamerať na vysvetlenie princípov komunikačných protokolov. Pre prvú úlohu bol vybraný smerovací protokol BGP a pre druhú protokol ICMPv6.

Prvá kapitola obsahuje teoretické znalosti na základe ktorých boli navrhované úlohy vytvorené.

Táto kapitola je rozdelená na tri časti. Prvé dve časti opisujú spôsob fungovania protokolu BGP a preštudované možnosti použitia protokolu BGP pre prostredie v simulátore ns-3. Týmito možnosťami boli dva voľne dostupné súkromné projekty a možnosť použitia protokolu BGP v rámci aplikácie Quagga. V každej z možností bolo odskúšané vytvorenie topológie pozostávajúcej z aspoň troch autonómnych systémov medzi ktorými bolo potrebné vytvoriť susedstvá a analyzovať ich prepojenie. Pre tvorbu laboratórnej úlohy bola vybraná možnosť s aplikáciou Quagga, ktorá ponúkala pre použitie protokolu BGP v prostredí ns-3 najvhodnejšie možnosti.

Tretia časť kapitoly sa zaoberá opisom správ protokolu ICMPv6 a jeho funkcionalít v rámci IPv6 sietí. Táto časť obsahuje opis použitých informačných a chybových správ protokolu. Z informačných správ to sú správy využívané pri konfigurácii IPv6 adries, podpore zmeny smerovania v sieti, zisťovaní dostupnosti zariadenia a získaní linkovej adresy zariadenia. Z chybových správ to sú správy vyvolané pri nedostupnosti zariadenia, pri znížení počtu hopov paketu na nulovú hodnotu a pri zasielaní príliš veľkého paketu.

Druhá kapitola práce obsahuje navrhnuté a vytvorené laboratórne úlohy.

V prvej úlohe je študentom opísané fungovanie a konfigurácia protokolu BGP pre výmenu smerovacích informácii medzi autonómnymi systémami. Pre študentov je na začiatku úlohy vytvorený súbor s kódom, kde je implementovaná základná topológia siete. V prvom kroku úlohy je popísaný spôsob implementácie základnej topológie, kam má študent na základe definovaného zadania doplniť vytvorenie jedného zo susedstiev. V druhom kroku, je uvedený spôsob pripojenia ďalšieho autonómneho systému k základnej topológii. V treťom kroku, je v topológii simulovaný výpadok linky a reakcia protokolu BGP naň. V štvrtom kroku, je ukázané, ako sa zmení trasa po nastavení tzv. PeerLinku medzi smerovačmi. Piaty, záverečný krok, tvorí samostatná práca študenta, v ktorej má do výslednej topológie pridať ďalší autonómny systém a overiť jeho správnu konfiguráciu.

Druhá úloha sa zaoberá štruktúrou a funkcionalitou správ protokolu ICMPv6. Táto úloha je rozdelená na tri časti. V prvej časti, sú popísané správy súvisiace s konfiguráciou IPv6 adries. V úlohe sú využité dva typy konfigurácii - statická a bezstavová autokonfigurácia. V druhej časti je využité statické smerovanie s neefektívnymi ces-

tami k vyvolaniu správy ICMPv6, ktorá toto smerovanie dokáže ovplyvniť, ďalej je v nej ukázaná funkcionalita protokolu ICMPv6 pre získanie MAC adresy z IPv6 adresy a správy spojené so zisťovaním dostupnosti zariadenia. V závere tejto časti je do úlohy pridaný protokol RIPng, ako ukážka spôsobu dynamického smerovania v rámci IPv6 sietí. Tretia časť úlohy je zameraná na chybové správy protokolu. Do úlohy sú postupne pridané tri chyby, ktoré v IPv6 sieti môžu nastať a je ukázaný spôsob akým na ne protokol ICMPv6 reaguje.

1 Teoretická časť práce

Teoretická časť tejto diplomovej práce sa venuje popisu protokolov BGP (Border Gateway Protocol) a ICMPv6 (Internet Control Message Protocol version 6), pre ktoré boli v praktickej časti vytvorené jednotlivé laboratórne úlohy. Táto časť taktiež obsahuje popis existujúcich možností konfigurácie protokolu BGP v simulátore ns-3, ktoré boli odskúšané pre návrh prvej laboratórnej úlohy.

1.1 Protokol BGP

Každé zariadenie komunikujúce cez Internet so zariadením v inej sieti, musí mať pre úspešné posielanie správ zabezpečené správne smerovanie medzi týmito sieťami. Internet je súborom obrovského množstva sietí a nie je možné aby si každý smerovač uchovával a aktualizoval informácie o všetkých ostatných sieťach. Internet je teda rozdelený do tzv. autonómnych systémov (ďalej ako AS) a smerovanie v takto veľkých sieťach prebieha na dvoch úrovniach a to v rámci jedného autonómneho systému, a medzi týmito autonómnymi systémami [1].
AS je množina smerovačov pod jednotnou správou používajúca Interior Gateway Protocol (IGP) k smerovaniu paketov v rámci jedného AS a Exterior Gateway Protocol (EGP) k smerovaniu paketov susedným AS [2].

1.1.1 IGP

Každá sieť má svoj adresný rozsah. AS sa teda skladá z istej množiny adresných rozsahov a sietí, ktoré tieto rozsahy definujú. Každý AS spravuje istá organizácia (napr. ISP) a každá zo sietí v danom AS spadá pod istý hraničný smerovač - vnútornú bránu daného AS (ďalej ako vnútorný smerovač). Všetky protokoly používané pre prenos informácii medzi týmito hraničnými smerovačmi sa označujú ako protokoly IGP, sú to napríklad RIPv2 (Routing Information Protocol), EIGRP (Enhanced Interior Gateway Routing Protocol), OSPF (Open Shortest Path First) alebo IS-IS (Intermediate System-to-Intermediate System). Tieto protokoly tvoria IGP stratégiu daného AS [1].
Dôležité je, že ostatným AS sa daný AS javí ako jednotný smerovací celok s konzistentným obrazom a informáciami o tom, aké siete sú cez neho dostupné [3].

1.1.2 EGP

Pre distribúciu smerovacích informácii medzi AS slúži protokol EGP. Každý AS by mal mať minimálne jeden hraničný smerovač, ktorý predáva potrebné smerovacie

informácie inému AS, je teda externou bránou daného AS (ďalej ako externý smerovač). Momentálne používaným EGP protokolom je protokol BGP vo verzii 4 [4], ktorý zaisťuje dynamické prepojenie AS a umožňuje externým smerovačom automaticky reagovať na zmeny v topológii sieti [1].
Externý smerovač jedného AS si po vytvorení spojenia s externým smerovačom druhého AS vymieňajú smerovacie informácie na základe ktorých si vytvárajú graf prepojení jednotlivých AS [2].
Toto spojenie sa nazýva tiež aj ako susedstvo. Susedstvá medzi jednotlivými externými smerovačmi sú konfigurované ručne administrátorom a pre prenos daných informácii je použitý protokol TCP (Transmission Control Protocol) na porte 179 [1].
Každý externý smerovač si vytvára smerovaciu tabuľku, ktorá v sebe obsahuje množinu všetkých existujúcich ciest, ktoré smerovač má a zoznam sietí, ktoré sú dostupné každou z týchto ciest. Cesta je zoznam AS, ktoré sú na trase k danej sieti. Aj keď si BGP udržiava všetky dostupné cesty, ďalej posiela len preferované [2].

1.1.3 BGP správy

BGP podporuje štyri typy správ - OPEN, KEEPALIVE, UPDATE a NOTIFICATION. Každá správa má hlavičku s pevnou dĺžkou 19B. Za hlavičkou môže, ale nemusí byť dátová časť podľa typu danej správy [2].
Hlavička správy sa skladá z troch častí [4]:

1. **Marker** - je položka nutná kvôli kompatibilite pre všetky verzie protokolu. Toto pole musí obsahovať samé jednotky.
2. **Length - Dĺžka** - udáva celkovú dĺžka správy spolu s hlavičkou. Dĺžka celej správy bude stále minimálne 19B (hlavička) a maximálne 4096B.
3. **Type - Typ** - je kód ktorý určuje typ správy:
   - 1 pre správu OPEN
   - 2 pre správu UPDATE
   - 3 pre správu NOTIFICATION
   - 4 pre správu KEEPALIVE

Správa OPEN

Správa OPEN je prvou správou, ktorú si obe strany pošlú hneď po ustanovení TCP spojenia. Jej minimálna veľkosť je 29B (spolu s hlavičkou) [4].
Správa k hlavičke pridáva nasledujúce polia [4]:

- **Version - Verzia** protokolu verziu protokolu správy.
- **My Autonomous System - Môj autonómny systém** je číslo AS odosielateľa.

- **Hold Time - Hold Time** udáva počet sekúnd, ktoré odosielateľ navrhuje ako hodnotu pre dobu Hold Time intervalu. Hodnota udáva maximálny počet sekúnd, koľko môže ubehnúť medzi prijatím po sebe idúcich KEEPALIVE a/alebo UPDATE správ od odosielateľa.
- **BGP Identifier - BGP identifikátor** je identifikátor odosielateľa. Hodnota BGP identifikátoru je daná na začiatku spojenia a je rovnaká pre každé lokálne rozhranie a BGP suseda.
- **Optional Parameters Length - Dĺžka voliteľných parametrov** udáva celkovú dĺžku poľa Optional Parameters - Voliteľné parametre v bytoch. Ak je táto hodnota nula, žiaden voliteľný parameter sa v nej nenachádza.
- **Optional Parameters - Voliteľné parametre** pole obsahuje zoznam voliteľných parametrov, kde každý parameter je zapísaný vo formáte <Typ parametru, Dĺžka parametru, Hodnota parametru>.

Správa UPDATE

Správy UPDATE sú použité pre prenos smerovacích informácii medzi susednými BGP smerovačmi. Informácie v tejto správe sú využité k tvorbe grafu z jednotlivých AS pre smerovanie medzi nimi. Uplatnením dohodnutých pravidiel sa podľa týchto informácii môžu detegovať a odstrániť smerovacie slučky a ďalšie anomálie pri smerovaní medzi AS [4].
Správa je využitá aj k propagácii dostupných trás alebo k odstráneniu niekoľkých nedostupných trás z obehu. Jedna UPDATE práva môže naraz propagovať dostupnú cestu a zároveň odstrániť z obehu niekoľko nedostupných ciest [4].
UPDATE správa okrem hlavičky taktiež obsahuje tieto polia (nie všetky sú stále v správe prítomné) [4]:

- **Withdrawn Routes Length - Dĺžka poľa trás k odstráneniu** udáva celkovú dĺžku poľa Withdrawn Routes v bytoch. Hodnota 0 naznačuje, že sa žiadna cesta z obehu neodstraňuje a pole Withdrawn Routes nie je potom v tejto UPDATE správe prítomné.
- **Withdrawn Routes - Trasy k odstráneniu** toto pole obsahuje zoznam IP adries s prefixmi pre trasy, ktoré majú byť odstránené z obehu.
- **Total Path Attribute Length - Celková dĺžka poľa atribútov ciest** udáva celkovú dĺžku poľa Path Attributes v bytoch.
- **Path Attributes - Atribúty cesty** je postupnosť atribútov danej cesty. Táto postupnosť sa nachádza v každej UPDATE správe okrem tej, ktorá slúži len k odstráneniu trasy z obehu.
- **Network Layer Reachability Information NLRI - Informácia o dostupnosti sietí** obsahuje zoznam prefixov IP adries. Informácia o dostupnosti

je uvedená vo forme dvojice <length, prefix>, kde:
  - Length udáva dĺžku prefixu IP adresy v bitoch. Ak je táto hodnota 0, prefix zodpovedá všetkým IP adresám.
  - Prefix obsahuje prefix IP adresy.

  Minimálna veľkosť správy UPDATE je 23B, z ktorých 19B je hlavička, ďalšie 2B patria Withdrawn Routes Length a ďalšie 2B Total Path Attribute Length [4].

Správa NOTIFICATION

Správa NOTIFICATION je zaslaná ak je na sieti detekovaný chybový stav. Po odoslaný tejto správy sa BGP spojenie ihneď uzatvára [4].
Správa k hlavičke pridáva nasledujúce polia [4]:

- **Error Code - Číslo chyby** udáva typ upozornenia. Definované sú tieto chybové čísla:
  - 1 - Message Header Error
  - 2 - OPEN Message Error
  - 3 - UPDATE Message Error
  - 4 - Hold Timer Expired
  - 5 - Finite State Maachine Error
  - 6 - Cease
- **Error subcode - Dotačné číslo chyby** udáva dodatočnú informáciu o vzniknutej chybe. Každá z chýb má jedno alebo viac dodatočných chybových čísel, ktoré s ňou súvisia. Ak nie je definované žiadne vhodné dodatočné číslo chyby, je použitá hodnota 0. Pre správu UPDATE sú definované dodatočné chybové čísla ako napríklad 1 - Nesprávny formát zoznamu atribútov, 2 - Nerozoznaný známy atribút atď.
- **Data** toto pole slúži k diagnostike príčiny správy NOTIFICATION. Obsah tohto poľa závisí na čísle a dodatočnom čísle chyby.

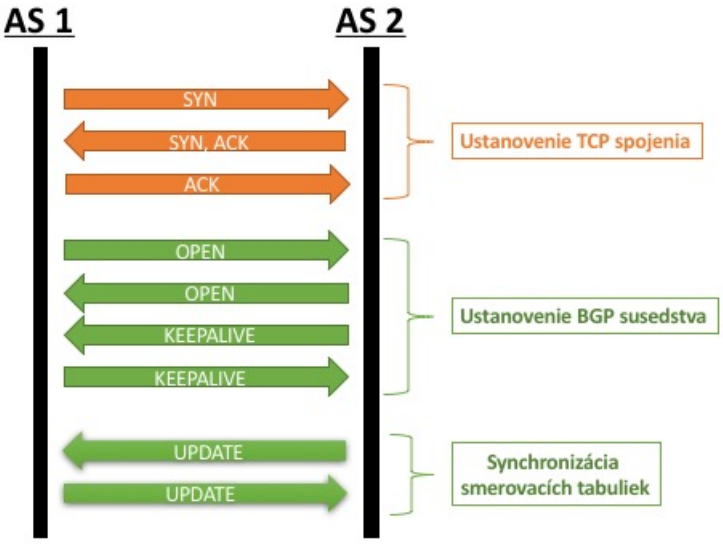
Minimálna veľkosť správy NOTIFICATION je 21 bytov, spolu s hlavičkou správy [4].

Správa KEEPALIVE

BGP nepoužíva k overeniu dostupnosti smerovačov keep-alive mechanizmus protokolu TCP. Namiesto neho si smerovače dostatočne často vymieňajú správu KEEPALIVE aby nevypršal interval Hold Time. Za odporúčanú dobu nastavenia výmeny správ KEEPALIVE sa považuje tretina intervalu Hold Time. KEEPALIVE správy nemôžu byť zaslané častejšie ako jedna za sekundu a implementácia protokolu si môže prispôsobiť frekvenciu zasielania KEEPALIVE správ. KEEPALIVE správa pozostáva len z hlavičky správy, má teda dĺžku 19B [4].

1.1.4 Vytvorenie spojenia

Prvou odoslanou správou po ustanovení TCP spojenia, je správa OPEN, ktorú si navzájom pošlú oba externé smerovače. Ak je správa OPEN akceptovaná, druhá strana pošle správu KEEPALIVE ako potvrdenie správy OPEN. To isté musí urobiť aj druhá strana. Ak jedna zo strán správu OPEN nepotvrdí, je odoslaná správa NOTIFICATION, po ktorej sa ihneď ukončí BGP spojenie [2].



Obr. 1.1: Výmena správ pri ustanovení BGP susedstva, prevzaté z [5].

Po potvrdení BGP spojenia si medzi sebou smerovače vymenia správy UPDATE, kde sú informácie o sieťach, ktoré sú pre nich dostupné. Po výmene UPDATE správ prechádza sieť do stáleho stavu. V stálom stave sa vymieňajú správy KEEPALIVE, pre udržanie aktívneho BGP spojenia. Môžu sa v ňom posielať aj doplňajúce UPDATE správy, ak sa topológia siete nejako zmení. Správa NOTIFICATION je vyslaná, ak v sieti nastane chyba alebo sa ukončí BGP spojenie. Po odoslaní správy NOTIFICATION sa BGP spojenie uzatvorí [2].

1.2 Možnosti konfigurácie protokolu BGP v simulátore ns-3

Simulátor ns-3

ns-3 je udalosťami riadený diskrétny sieťový simulátor zameraný predovšetkým na výskum a vzdelávanie. Je to program napísaný v programovacom jazyku C++, ktorý je voľne dostupný pod licenciou GNU GPLv2 a je verejne dostupný pre výskum, vývoj a použitie [6].
Cieľom projektu ns-3 je vývoj a uprednostňovanie voľne dostupného prostredia pre výskum sietí. ns-3 má byť čo najviac modifikovateľný a rozšíriteľný, pričom pre užívateľov ponúka už vopred vytvorené príklady modelov simulácii, no očakáva sa, že väčšina užívateľov bude vytvárať nové modely, prípadne už vytvorené modely ďalej modifikovať a skúšať v rôznych konfiguráciách [6].
ns-3 ponúka podporu pre [6]:

- Vytváranie virtuálnych sietí (uzlov, kanálov, aplikácií) a podporu pre prvky ako plánovač udalostí, generátory topológii, časovače, náhodné premenné a ďalšie objekty pre podporu diskrétnej, udalosťami riadenej sieťovej simulácie zameranej na internetovú a ďalšie možné paketovo-orientované sieťové systémy.
- Podporu sieťovej emulácie - možnosť pre procesy simulátoru k vytváraniu a využitiu reálnych sieťových paketov.
- Podporu distribuovanej simulácie - možnosť aby boli simulácie distribuované medzi viaceré procesory či stroje.
- Podporu pre animáciu sieťových simulácii
- Podporu pre sledovanie, logovanie a výpočet štatistík z výstupu simulácie

ns-3 má modulárnu implementáciu, ktorá obsahuje tzv. *core* knižnicu, tá zabezpečuje podporu všeobecných aspektov simulátoru ako sú debugovacie objekty, generátory náhodných čísel, smart pointry, podporu pre callback funkcie, unit testy a zoznam referencii a *simulator* knižnicu definujúcu objekty pre prácu s časom, plánovače a udalosti. Všeobecná *common* knižnica definuje objekty, ktoré sú nezávislé na špecifických sieťových architektúrach, ako napríklad pakety a trasovanie objekty. Dôležitou je *node* knižnica. Táto knižnica základné abstraktné triedy pre úplny základ objektov v simulátore, ako sú napríklad uzly, kanály a sieťové zariadenia. Internetovo zamerané modely (IP a transportné protokoly) sa nachádzajú v *internet-node* knižnici. Špecifické zariadenia ako napríklad Ethernet sú v knižnici *device*. Užívatelia si môžu napísať vytvoriť a pridať aj vlastné knižnice. Pričom modulárna implementácia dovoľuje kompiláciu aj pre samostatné menšie časti a spustiteľné ns-3 programy môžu byť zostavené do bud staticky alebo dynamicky nalinkovaných knižníc [6].

1.2.1 Preskúmané možnosti

V súčasnej verzii simulátor ns-3.32, ani v starších verziách, neponúka priamo podporu pre konfiguráciu protokolu BGP, avšak modulárna implementácia umožňuje používateľom vytvárať vlastné možnosti podpory pre tento protokol.
V rámci diplomovej práve boli preskúmané tieto tri možnosti podpory protokolu BGP:

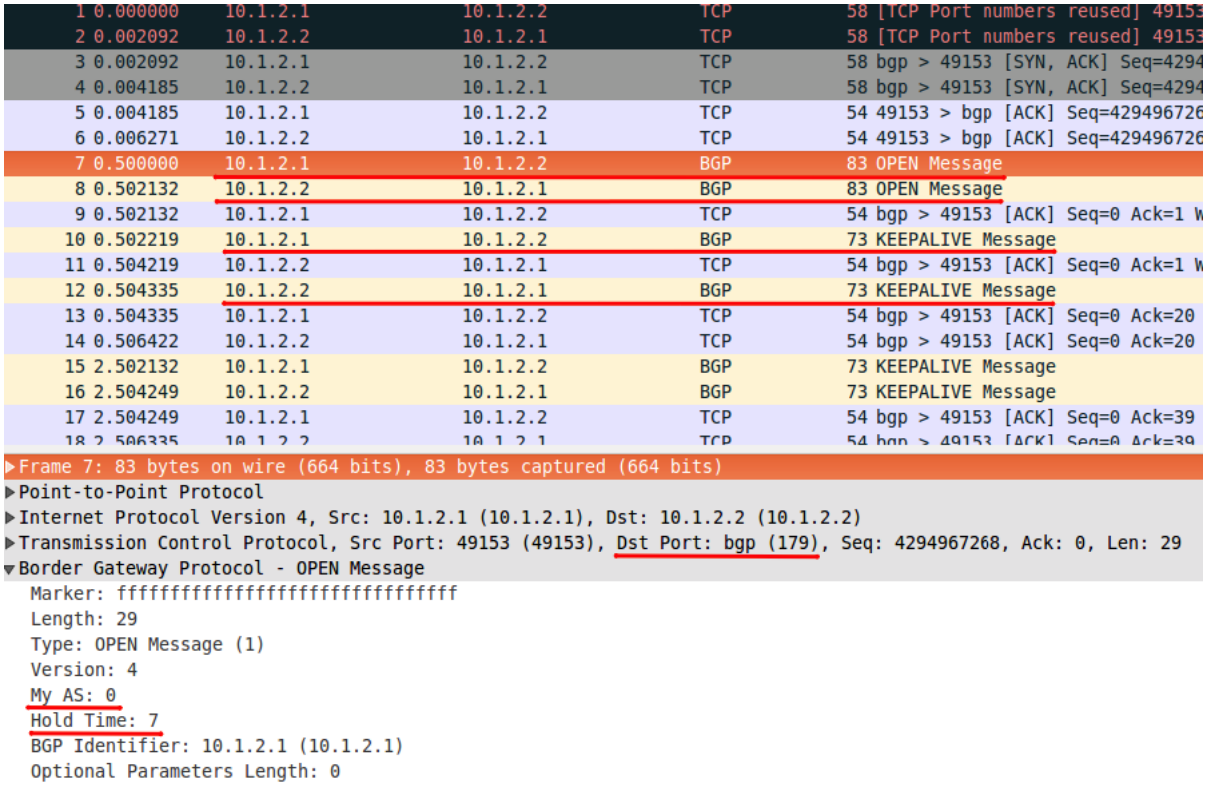
- ns-BGP [7]
- Nat-Lab/ns-bgp [8]
- DCE a Quagga [9]

ns3-BGP

Táto možnosť podpory protokolu BGP do simulátoru ns-3 bola vytvorená ako projekt, ktorý vypracoval M.R.Sahraei v roku 2012 [7]. Projekt bol vytvorený pre verziu ns-3.13, no bol kompatibilný aj s verziou ns-3.21, na ktorej bol v rámci tohto projektu odskúšaný.
Projekt ns3-BGP obsahuje všetky potrebné zdrojové súbory, ktoré stačí nakopírovať na správne miesto uvedené v návode. Tieto súbory obsahujú aj príklad použitia vytvorených tried, inštalácia si teda nevyžaduje žiaden ďalší software. Je však potrebné si dať pozor na správne umiestnenie daných súborov a úpravu skriptu `wscript`, kde je uvedené ako sa má daný súbor s príkladom prekladať.
V rámci projektu M.R.Sahraei vytvoril triedy `bgp-router.cc`, `bgp-router-helper.cc` a `bgp-pdu.cc` [7], pomocou ktorých je možné nakonfigurovať a vytvoriť aplikáciu pre uzol, ktorá sa na danom uzle správa podobne ako hraničný BGP smerovač.
Obrázok 1.2 zobrazuje vytvorenie susedstva medzi dvoma smerovačmi za pomoci týchto tried. Na obrázku 1.2 je vidieť, že smerovaču je možné priradiť číslo jeho AS a hodnotu pre interval Hold Time. To, že sa dané susedstvo vytvorilo potvrdzuje aj vytvorený trasovací súbor 1.3, kde je vidieť, že smerovače si vymenili správy OPEN a KEEPALIVE. Paket bol odoslaný na správy port 179 pre protokol BGP, bol správne uvedené číslo AS a tiež aj hodnota, ktorú daný smerovač navrhoval ako interval pre Hold Time. V rámci skúšania možností tejto možnosti podpory protokolu boli vytvorené ďalšie dve susedstvá, ktorých vytvorenie prebehlo taktiež úspešne. Avšak projekt ns3-BGP nemá kompletnú implementáciu fungovania protokolu BGP a chýba mu v ňom implementácia správ UPDATE a NOTIFICATION, k výmene smerovacích informácii a detekcii výpadkov a možnosti nastavenia atribútov protokolu.

```
//BGP settings
                                  // local address,            remote address
BgpRouterHelper cBgpRouterHelper0 (ipv4InterfacesP2P.GetAddress(0), ipv4InterfacesP2P.GetAddress(1));
cBgpRouterHelper0.SetAttribute ("LocalASNumber", UintegerValue (0));
cBgpRouterHelper0.SetAttribute ("HoldTime", UintegerValue (7));

ApplicationContainer bgpRouterApps0 = cBgpRouterHelper0.Install(p2p.Get(0));
bgpRouterApps0.Start (Seconds (0.0));
bgpRouterApps0.Stop (Seconds (10.0));


BgpRouterHelper cBgpRouterHelper1 (ipv4InterfacesP2P.GetAddress(1), ipv4InterfacesP2P.GetAddress (0));
cBgpRouterHelper1.SetAttribute ("LocalASNumber", UintegerValue (1));
cBgpRouterHelper1.SetAttribute ("HoldTime", UintegerValue (10));

ApplicationContainer bgpRouterApps1 = cBgpRouterHelper1.Install(p2p.Get(1));
bgpRouterApps1.Start (Seconds (0.0));
bgpRouterApps1.Stop (Seconds (10.0));
```

Obr. 1.2: Nastavenie susedstva smerovačov v ns3-BGP



Obr. 1.3: Trasovací súbor obsahujúci pakety vytvorenia susedstva smerovačov v ns3-BGP

Nat-Lab/ns-bgp

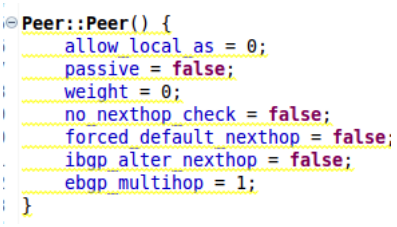
Ďalšiu možnosť ako použiť protokol BGP v simulátore ns-3 vytvoril Nato Morichika ako svoj osobný projekt. Ten je voľne dostupný na odkaze [8] pod licenciou MIT. Táto implementácia pracuje s knižnicou `libbgp`, napísanou v jazyku C++. Táto knižnica obsahuje všetky potrebné infraštruktúry k vytvoreniu BGP smero-

vača [10].
Inštalácia tejto možnosti je jednoduchá, je však potrebné si dať pozor na verziu prekladača `gcc/g++`, ktorý je použitý pri preklade ns-3. V rámci tohto projektu bola odskúšaná inštalácia na verzii ns-3.21, kde nebola úspešná a na verzii ns-3.30, kde prebehla podľa úspešne podľa návodu uvedeného v [8]. V tomto repozitári [8] je uvedený aj jednoduchý príklad vytvorenia susedstva dvoch smerovačov. Na základe priloženého príkladu bola vytvorená aj jednoduchá skúšobná topológia pre vyskúšanie tejto implementácie protokolu BGP pozostávajúca z 3 smerovačov, pričom 2 z nich boli hraničnými smerovačmi v AS a tretí reprezentoval sieť vo vnútri jedného z AS.
Obrázok 1.4 zobrazuje ako sa v tejto implementácii vytvára susedstvo smerovačov. Pri definovaní susedstva je možné nastaviť číslo AS, IP adresu pre suseda a ďalšie parametre, ktoré sú uvedené v triede `Peer`, obr. 1.5.

```
PointToPointHelper pointToPoint;
NetDeviceContainer n0n1 = pointToPoint.Install(p2pNodes.Get(0), p2pNodes.Get(1));
NetDeviceContainer n0n2 = pointToPoint.Install(p2pNodes.Get(0), p2pNodes.Get(2));

Ipv4AddressHelper ipv4;
ipv4.SetBase("10.0.0.0", "255.255.255.0");
Ipv4InterfaceContainer i0i1 = ipv4.Assign(n0n1);

ipv4.SetBase("10.1.0.0", "255.255.255.0");
Ipv4InterfaceContainer i0i2 = ipv4.Assign(n0n2);

//R0 -> peer -> R1
Ptr<Bgp> bgp_app_1 = CreateObject<Bgp>();
bgp_app_1->SetAttribute("RouterID", Ipv4AddressValue(i0i1.GetAddress(0)));
bgp_app_1->SetAttribute("LibbgpLogLevel", EnumValue(libbgp::DEBUG));
Peer bgp_app_1_peer;
bgp_app_1_peer.peer_address = i0i1.GetAddress(1);
bgp_app_1_peer.peer_asn = 65001;
bgp_app_1_peer.local_asn = 65000;
bgp_app_1_peer.passive = false;
bgp_app_1->AddPeer(bgp_app_1_peer);
p2pNodes.Get(0)->AddApplication(bgp_app_1);

//R0 <- peer <- R1
Ptr<Bgp> bgp_app_2 = CreateObject<Bgp>();
bgp_app_2->SetAttribute("RouterID", Ipv4AddressValue(i0i1.GetAddress(1)));
Peer bgp_app_2_peer;
bgp_app_2_peer.peer_address = i0i1.GetAddress(0);
bgp_app_2_peer.peer_asn = 65000;
bgp_app_2_peer.local_asn = 65001;
bgp_app_2_peer.passive = false;
bgp_app_2->AddPeer(bgp_app_2_peer);
p2pNodes.Get(1)->AddApplication(bgp_app_2);

bgp_app_1->SetStopTime(Seconds(100));
bgp_app_2->SetStopTime(Seconds(100));
```

Obr. 1.4: Nastavenie susedstva v Nat-Lab/ns-bgp

Vytvorenie susedstva je možné zachytiť vo vytvorenom trasovacom súbore, ktorého odfiltrovaný BGP obsah je zobrazený na obrázku 1.6. Smerovače si navzájom zaslali správy typu OPEN a následne aj správy typu KEEPALIVE. Z neznámeho dôvodu, si v tejto implementácii smerovače vymenili aj správy typu NOTIFICATION, pri-



Obr. 1.5: Úsek z definície parametrov triedy Peer v Nat-Lab/ns-bgp

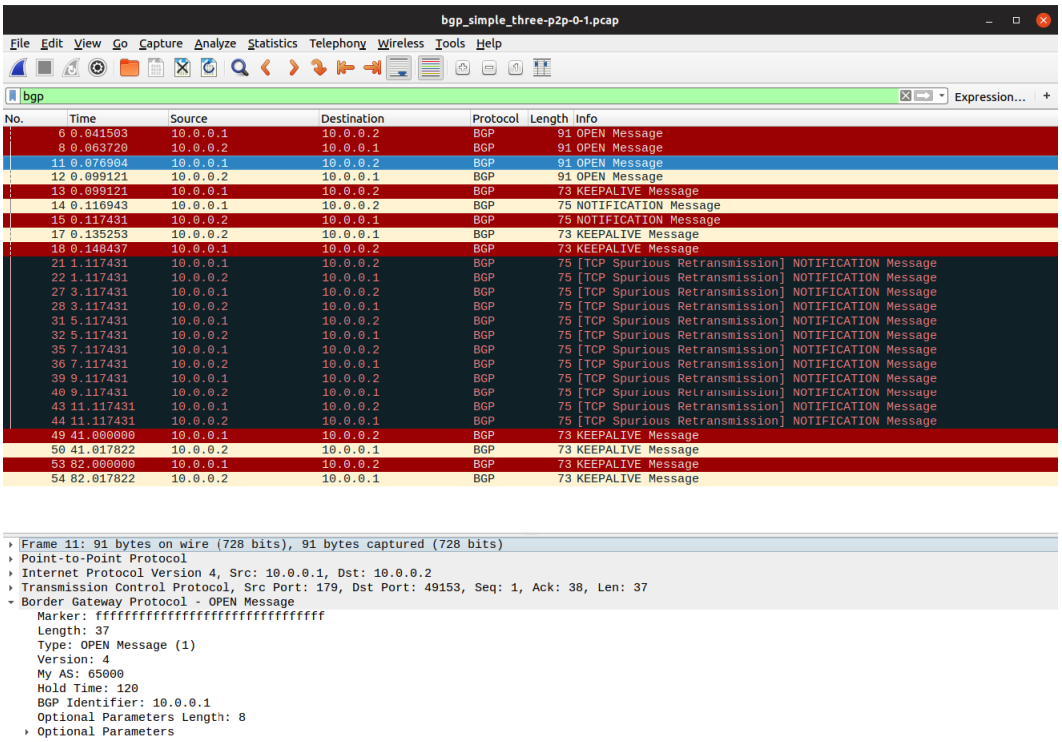
čom po tejto správe by malo byť BGP spojenie ukončené. Ako je však vidieť v dolnej časti obrázku 1.6, smerovače si neskôr zasa navzájom vymenili správy typu KEEPALIVE, čo značí, že spojenie bolo aktívne. Ďalšou anomáliou tejto implementácie je, že smerovače by si po vytvorení susedstva mali poslať správu typu UPDATE s informáciou o dostupných sieťach. Aj keď táto možnosť použitia protokolu BGP správy UPDATE podporuje, spôsob ich vyvolania je odlišný. Pre ich odoslanie je potrebné v kóde pridať vytvorenej BGP aplikácii informáciu o tom, aké cesty sú z tohto smerovača dostupné pomocou funkcie `AddRoute()`. V rámci skúšania možností tejto implementácie, bol do tejto jednoduchej topológie ešte pridaný aj ďalší AS, pričom ani pridanie ďalšieho susedstva správu UPDATE nevyvolalo.
Táto možnosť použitia protokolu BGP s ns3 bola vytvorená v roku 2019 a jej autor je stále aktívny pri dolaďovaní jej fungovania. To by mohlo v budúcnosti znamenať upravenie jej zdokonalenie a pridanie ďalších parametrov protokolu BGP.

DCE a Quagga

Treťou možnosťou pre prácu s protokolom BGP v simulátore ns-3 je pridanie modulu priameho vykonávania kódu (DCE - Direct Code Execution) s aplikáciou Quagga. Priame vykonávanie kódu je modul, ktorý ponúka možnosť v prostredí ns-3 využiť už existujúce implementácie sieťových protokolov alebo aplikácii z užívateľského priestoru a priestoru jadra systému, bez nutnosti zásahu do zdrojového kódu. To napríklad umožňuje namiesto aplikácie `V4PingHelper`, vytvorenej v ns-3, použiť skutočný `ping`. Tento modul je možné využiť v dvoch módoch [11]:

1. Základný mód, v ktorom ňom modul využíva TCP stack prostredia ns-3
2. Pokročilý mód, v ktorom modul využíva sieťový Linux stack.

Pre tento modul bolo vytvorených viacero aplikácii pre ich použitie v prostredí ns-3, ktoré ho využívajú - sú to napríklad CCNx, iperf či ping. Ďalšou z nich je aplikácia Quagga. Quagga umožňuje užívateľom v simuláciách použiť systémovú implementáciu smerovacích protokolov ako RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP, BGP+, RAadv. Pre túto prácu je dôležitá hlavne podpora protokolu BGP, ktorý Quagga ponúka pre obe módy modulu priameho vykonávania kódu.



Obr. 1.6: Trasovací súbor obsahujúci pakety vytvorenia susedstva smerovačov v Nat-Lab/ns-bgp

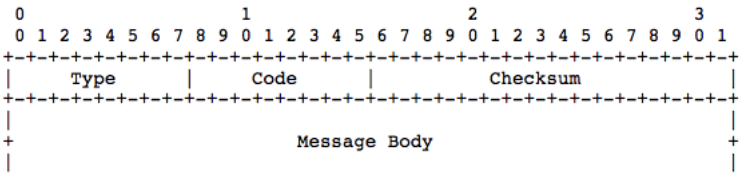
Inštalácia DCE a aplikácie Quagga je popísaná v návode [9], kde je nutné dbať na správne nainštalovanie balíčkov, ktoré DCE a Quagga vyžadujú a pri preklade DCE je nutné uviesť v akom móde bude modul využitý. Pre účely tejto práce bol zvolený základný mód, ktorý bol s aplikáciou nainštalovaný vo verzii simulátoru ns-3.21.
Táto možnosť použitia protokolu BGP v prostredí ns-3 bola zvolená aj pre účel vytvorenia laboratórnej úlohy v rámci tejto semestrálnej práce. Úloha má demonštrovať konfiguráciu a fungovanie protokolu BGP študentom. K tomuto účelu aplikácia podporuje všetky typy správ protokolu, vytváranie viacerých AS a korektné prepojenie ich hraničných smerovačov. Bohužiaľ, ani Quagga neponúka možnosť voľby parametrov BGP pre úpravu smerovacej politiky na úrovni AS ako sú napríklad MED, či LOCAL_PERF. Ukážku vytvorenia susedstva a možností práce s aplikáciou Quagga je možné vidieť v kapitole 2.1, kde je uvedený celý návod pre vytvorenú laboratórnu úlohu.

1.3 Protokol ICMPv6

Internet Control Message Protocol (ICMP) je režijným protokolom Internetu. Slúži k ohlasovaniu chybových stavov, testovaniu dosiahnuteľnosti a všeobecne k výmene niektorých informácií o prevádzke v sieti. Jeho implementácia je povinná na každom zariadení podporujúcom IP [12].
Internet Control Message Protocol version 6 (ICMPv6) je implementáciou protokolu ICMP pre siete využívajúce protokol IPv6 a oproti pôvodnej implementácii prináša niekoľko zmien [13].
ICMPv6 pokrýva funkcionalitu ICMP, no zároveň v sebe kombinuje aj funkcie, ktoré boli pred jeho nástupom rozdelené medzi rôznymi protokolmi, napríklad funkcionalitu protokolu IGMP (Internet Group Membership Protocol) a ARP (Address Resolution Protocol). Zavádza taktiež niektoré zjednodušenia tým, že eliminuje zastarané typy správ, ktoré sa už nepoužívajú [14].
Najviac sa protokoly ICMP a ICMPv6 líšia v konkrétnych typoch používaných správ a ich formátoch [15].
Každej správe protokolu ICMPv6 predchádza hlavička protokolu IPv6 a prípadne žiadna alebo viac rozšírených hlavičiek IPv6 [13]. To, že IP datagram nesie ICMPv6 správu signalizuje v položke Hext Header hodnota 58 [12].

Formát ICMPv6 správ

Všeobecne majú ICMPv6 správy nasledujúci formát, ktorý obsahuje polia [13]:



Obr. 1.7: Formát správy ICMPV6, prevzaté z [13].

- **Type - Typ** je hodnota, ktorá určuje formát dát, aké správa obsahuje.
- **Code - Kód** závisí na type správy. Používa sa pre pridanie detailov správy.
- **Checksum - Kontrolný súčet** sa používa k detekcii poškodenia dát správy a častí IPv6 hlavičky.
- **Message Body - Telo správy**, ktoré obsahuje samotné dáta správy.

ICMPv6 správy sa delia na dve skupiny: chybové správy a informačné správy. Chybové správy majú hodnoty typu správ od 0 do 127 a informačné správy od 128 do 255 [13].

RFC 4443 [13] definuje formát pre nasledujúce správy:

- **Chybové správy ICMPv6**: Destination Unreachable (Typ 1), Packet Too Big (Typ 2), Time Exceeded (Typ 3), Parameter Problem (Typ 4), Private experimentation - (Typ 100 a 101) a Reserva pre rozšírenie chybových správ ICMPv6 (Typ 127).
- **Informačné správy ICMPv6**: Echo Request (Typ 128), Echo Reply (Typ 129), Private experimentation (Typ 200 a 201) a Reserva pre rozšírenie informačných správ ICMPv6 (Typ 255).

Navrhnutá laboratórna úloha pre protokol ICMPv6 je rozdelená na 3 časti. Prvé dve časti využívajú a opisujú informačné správy a posledná tretia časť sa venuje chybovým správam protokolu ICMPv6.

1.3.1 Informačné správy

Prvá časť úlohy pre protokol ICMPv6 (2.2.2) sa zaoberá dvoma spôsobmi konfigurácie IPv6 adries - statickou a bezstavovou. V rámci tejto časti, sú využité a opísané správy Neighbor Solicitation, Neighbor Advertisement, Router Solicitation a Router Advertisement.
V druhej časti (2.2.4) je ukázaná možnosť, ako môže protokol ICMPv6 pomôcť v rámci statického smerovania, ako nahradzuje funkcionalitu protokolu ARP v rámci IPv6 a príklad dynamického smerovania pomocou protokolu RIPng. Táto časť opisuje a využíva správy Redirect, ďalšiu funkcionalitu dvojice správ Neighbor Solicitation/Neighbor Advertisement, a správy Echo Request a Echo Reply.
Ako bolo spomenuté na začiatku tejto kapitoly, protokol ICMPv6 pokrýva taktiež funkcionality protokolov IGMP a ARP. K tomuto účelu boli vytvorené rozšírenia protokolu ICMPv6 ako napríklad Multicast Listener Discovery (MLD) pre protokol IGMP a Neighbor Discovery (ND) pre protokol ARP. V rámci prvých dvoch častí vytvorenej úlohy bolo využité rozšírenie Neighbor Discovery.

Neighbor Discovery

Toto rozšírenie, označované tiež aj ako Neighbor Discovery Protocol (NDP), je používané uzlami k určeniu adries susedných uzlov na linkovej vrstve, k nájdeniu susedných smerovačov, ktoré sú ochotné posielať ďalej ich pakety, k aktívnemu sledovaniu dostupnosti susedných uzlov a k detekcii zmien adries na linkovej vrstve [16].
ND definuje 5 rôznych ICMPv6 typov správ - dvojicu správ *Router Solicitation* a *Router Advertisement*, dvojicu správ *Neighbor Solicitation* a *Neighbor Advertisement* a správu *Redirect*. Tieto správy majú podobný formát ako ostatné správy, no pridávajú k nemu aj ďalšie polia, špecifické pre ich funkciu. K základným poliam - **Typ**, **Kód** (všetky ND správy majú definovaný len východzí kód s hodnotou 0, ktorý

nenesie žiadnu ďalšiu špecifikáciu) a **Kontrolný súčet** pridávajú polia **Reserved (Rezervované)** a **Options (Voľby)** [16].
Pole **Reserved - Rezervované** je zatiaľ nevyužité a pre každú správu musí byť odosielateľom inicializované na nulovú hodnotu a ignorované príjemcom. V poli **Options - Voľby** sú informácie špecifické pre daný typ správy, týchto polí môže správa obsahovať žiadnu alebo aj viac volieb a pre každý typ správy sú definované aj rôzne druhy volieb podľa ich obsahu. Všetky voľby obsahujú polia [16]:

- **Type - Typ** definuje typ danej voľby. Typy - *Source Link-Layer Address*, *Target Link-Layer Address*, *Prefix Information*, *Redirected Header* a *MTU*.
- **Length - Dĺžka** je dĺžka voľby. Dĺžka voľby s nulovou hodnotou je neplatná. Uzly musia ND paket s takouto dĺžkou voľby potichu zahodiť.

Správa RS - Router Solicitation

Táto správa je zasielaná uzlom po štarte svojho rozhrania a uzol ňou žiada smerovače o zaslanie správy RA, ešte pred zaslaním RA správy v naplánovanom čase. Pre **typ správy RS** je definovaná hodnota **133** [16].
V poli **Options - Voľby** môže správa obsahovať[16]:

- **Source link-layer address**, kde je uvedená linková adresa odosielateľa tejto správy, ak je táto adresa známa. Toto pole nie je prítomné ak je zdrojovou adresou nešpecifikovaná adresa.

Zdrojovou adresou tejto správy je buď adresa, ktorá bola pridelená rozhraniu odkiaľ bol správa odosielaná alebo ak rozhranie ešte nemalo pridelenú žiadnu adresu, tak je to nešpecifikovaná adresa ('::'). Cieľovou adresou zvyčajne býva adresa multicastovej skupiny pre všetky smerovače ('ff02::2') [16].

Správa RA - Router Advertisement

Smerovače v tejto správe posielajú do siete informácie o svojej existencii spolu s rôznymi parameterami siete. Správy RA obsahujú prefixy, ktoré rozhodujú o tom, či sa dané adresa nachádza v danej sieti, informácie pre konfiguráciu adries, navrhovanú hodnotu pre počet hopov apod. Tieto správy môžu byť zasielané periodicky alebo ako odpoveď na správu RS. Pre **typ správy RA** je definovaná hodnota **134** a okrem základných polí ďalej obsahuje polia [16]:

- **Cur Hop Limit** obsahuje východziu hodnotu, ktorá by mala byť umiestnená v poli Hop Count pre počet skokov v IP hlavičke pre odchádzajúce IP pakety. Nulová hodnota znamená (týmto smerovačom) nešpecifikovanú hodnotu.
- **Príznak M - Managed address configuration**, kde nastavenie tohto príznaku znamená, že uzly si môžu požiadať o adresu pomocou protokolu DHCPv6.

Nasledujúci príznak O je pri jeho nastavení redundantný a môže byť ignorovaný, pretože DHCPv6 poskytne všetky dostupné konfiguračné údaje.

- **Príznak O - Other configuration**, kde nastavenie tohto príznaku znamená, že ďalšie konfiguračné informácie sú k dispozícii pomocou DHCPv6. Sú to napríklad informácie o DNS alebo o ďalších serveroch v rámci siete. Ak nie je nastavený žiaden z príznak M ani O, znamená to, že žiadne z konfiguračných údajov nie sú dostupné pomocou DHCPv6.
- **Router Lifetime - Doba platnosti informácii smerovača** je doba platnosti informácii (v sekundách) priradená východziemu smerovaču. Nulová hodnota znamená, že smerovač nie je východzím smerovačom tejto siete. Možnosti, ktoré potrebujú časový limit pre ich informácie, obsahujú svoju vlastnú dobu platnosti.
- **Reachable Time - Doba dostupnosti** je čas v milisekundách, počas ktorého bude uzol predpokladať dostupnosť susedného uzlu po tom, čo obdržal potvrdenie o jeho dostupnosti.
- **Retrans Timer - Časovač preposielania** čas v milisekundách medzi preposielaním NS/NA správ.

Pre správu RA sú v RFC 4861 [16] definované tieto **Options (Voľby)**:

- **Source link-layer address** je linková adresa rozhrania, z ktorého bola správa RA odoslaná.
- **MTU**, kedy správa s definovanou voľbou MTU by mala byť posielaná na spojeniach s premenlivou veľkosťou MTU.
- **Prefix Information**, táto voľba špecifikuje prefixy, ktoré sú dostupné pre dané spojenia a môžu byť použité pre bezstavovú autokonfiguráciu. Smerovač by mal poskytovať všetky prefixy, ktoré považuje sa dostupné, aby multihome uzly mali kompletné informácie o dostupných cieľoch pre všetky spojenia ku ktorým sa pripájajú. Ak by uzlu s viacerými rozhraniami chýbali kompletné informácie, mohla by nastať možnosť, že si nebude môcť vedieť vybrať správne rozhranie pre posielanie dát.

Zdrojovou adresou tejto správy musí byť link-local adresa pridelená rozhraniu, z ktorého bola odoslaná. Cieľovou adresou je zvyčajne zdrojová adresa, z ktorej bola odoslaná správa RS alebo adresa multicastovej skupiny všetkých uzlov [16].

Správa NS - Neighbor Solicitation

Správa je zasielaná uzlom pre získanie linkovej adresy susedného uzla. V tejto správe uzol zasiela pre daný cieľ aj informáciu o svojej linkovej adrese. Správa tiež slúži k overeniu, či je susedný uzol stále dosiahnuteľný cez linkovú adresu zapísanú vo svojej vyrovnávacej pamäti a v mechanizme Duplicate Address Detection (DAD) (1.3.2).

Správy NS sú multicastové, ak uzol potrebuje zistiť linkovú adresu a unicastové, ak si uzol chce overiť dostupnosť susedného uzlu [16]. Pre **typ správy NS** je definovaná hodnota **135**. Tento typ správy obsahuje len základné polia a v poli **Options (Voľby)** môže obsahovať typy:

- **Source link-layer address**, kde je uvedená linková adresa odosielateľa tejto správy, ak je táto adresa známa. Toto pole nie je prítomné ak je zdrojovou adresou nešpecifikovaná adresa ('::').

Zdrojovou adresou je v tejto správe buď adresa pridelená rozhraniu, z ktorého je táto správa odosielaná alebo nešpecifikovaná adresa (ak sa jedná o proces DAD). Cieľovou adresou je buď multicastová solicited-node adresa patriaca k adrese cieľa alebo priamo adresa cieľa [16].

Správa Neighbor Advertisement NA

Správa je odpoveďou na správu NS. Uzol môže zaslať správu NA aj k oznámeniu zmeny linkovej adresy. Pre **typ správy NA** je definovaná hodnota **136** a okrem základných polí obsahuje polia [16]:

- **Príznak R - Router**, nastavenie tohto príznaku indikuje, že odosielateľom správy je smerovač.
- **Príznak S - Solicited**, nastavenie tohto príznaku indikuje, že správa NA bola zaslaná ako odpoveď na správu NS.
- **Príznak O - Override**, nastavenie tohto príznaku indikuje, že toto oznámenie (advertisement) by malo prepísať existujúci záznam o uzle a aktualizovať zapísanú linkovú adresu.
- **Target Address** pole obsahuje linkovú adresu uzlu, pre ktorý bola správa NS žiadaná v prípade ak bola táto správa zaslaná ako odpoveď na správu NS. Ak nebola táto správa zaslaná ako odpoveď pre NS, toto pole obsahuje linkovú adresu, ktorá sa zmenila.

Pre správu NA je definovaná táto **Option (Voľba)**:

- **Target link-layer address**, ktorá obsahuje linkovú adresu odosielateľa tejto správy.

Zdrojovou adresou tejto správy je adresa rozhrania, z ktorého bola zasielaná. Pre správu NA, ktorá je zaslaná ako odpoveď na správu NS, je cieľovou adresou adresa uzlu, ktorá zasielala správu NS alebo ak bola správa odoslaná s nešpecifikovanou adresou (pri procese DAD), je to adresa multicastovej skupiny všetkých uzlov, to platí aj ak nebola správa NA odoslaná ako odpoveď na správu NS [16].

Správa Redirect

Je zasielaná smerovačmi k informovaniu uzlu o existencii lepšej voľby ďalšieho kroku (next-hop) cesty k danému cieľu. Pre **typ správy RA** je definovaná hodnota **137** a okrem základných polí ďalej obsahuje polia [16]:

- **Target Address**, ktoré obsahuje IP adresu, ktorá bude lepšou adresou ďalšieho skoku (next-hop adresou) pre cieľovú adresu uvedenú v poli Destination Address.
- **Destination Address**, kde je IP adresa cieľa, pre ktorý je cesta zmenená adresou v poli Target Address.

Pre správu Redirect sú definované tieto **Options (Voľby)**:

- **Target link-layer address**, táto voľba obsahuje linkovú adresu pre Target Address.
- **Redirected Header**, táto voľba obsahuje čo najviac informácii o pakete, ktorý vyvolal zaslanie Redirect správy bez toho, aby prekročil stanovené MTU (Maximum Transmission Unit).

Zdrojovou adresou tejto správy musí byť link-local adresa pridelená rozhraniu, z ktorého bola zaslaná. Cieľovou adresou je zdrojová adresa uvedená v pakete, ktorý toto presmerovanie vyvolal [16].

1.3.2 Spôsoby konfigurácie adries pre IPv6

V prvej časti úlohy pre protokol ICMPv6 (2.2.2) sú popísané a ukázané dva spôsoby konfigurácie IPv6 adries - statická a bezstavová. Správy popísané v sekcii Neighbor Discovery (1.3.1) poskytujú týmto konfiguráciám ich funkcionalitu.

Statická konfigurácia a Duplicate Address Detection

Statická konfigurácia adresy je vo väčšine prípadov vykonávaná užívateľom alebo administrátorom. Môže však nastať situácia, že bude zariadeniu staticky nakonfigurovaná adresa, ktorá už v danej sieti existuje.
V rámci IPv6 je preto definovaný mechanizmus DAD - Duplicate Address Detection, ktorý by mal túto situáciu detegovať. Procedúra pre detekciu duplikátov adries využíva správy NS a NA. Ak je počas procedúry duplikátna adresa zistená, táto adresa nemôže byť rozhraniu pridelená. Ak adresa vznikla z identifikátoru rozhrania, musí byť rozhraniu pridelený nový identifikátor, alebo všetky IP adresy tohto budú musieť byť nakonfigurované manuálne [17].

DAD - Duplicate Address Detection

Mechanizmus DAD, pre overenie unikátnosti adresy v sieti, nastáva ak [18]:

- uzol spúšťa svoje rozhranie - kontrolujú sa samo-konfigurované link-local a pridelené statické adresy
- na rozhranie príde správa Router Advertisement s informáciou o novom prefixe s nastaveným príznakom pre použitie s bezstavovou konfiguráciou
- je vygenerovaná dočasná samo-konfigurovaná adresa

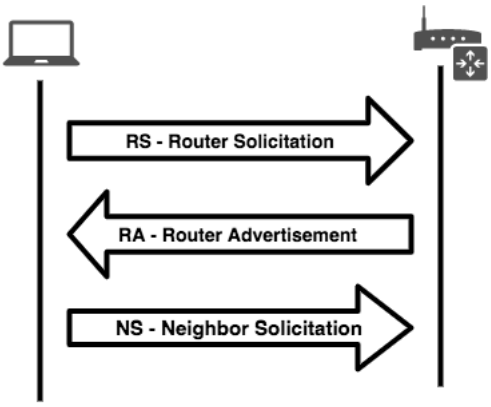
Proces DAD pozostáva z nasledujúcich krokov [18]:

1. Pri spustení rozhrania sa uzol pridáva k link-local multicastovej skupine pre všetky uzly.
2. Pridáva sa tiež do skupiny solicited-node pre adresu, ktorú si chce prideliť.
3. Je zaslaná správa Neighbor Solicitation na multicastovú adresu skupiny všetkých uzlov s adresou, ktorú si chce uzol prideliť a pre ktorú sa DAD vykonáva.
4. Uzol po istú dobu čaká na odpoveď od ktoréhokoľvek susedného uzlu - správu Neighbor Advertisement alebo Neighbor Solicitation.
5. Ak za túto dobu nepríde žiadna odpoveď - adresa je považovaná za unikátnu a rozhranie si ju môže priradiť.
6. Ak nejaká odpoveď príde - adresa nie je unikátna. Uzol opúšťa skupinu solicited-node pre danú adresu, vyvolá konzolovú správu 'Duplicated address detected' a označí túto adresu za nedostupnú.

DAD musí byť vykonané pre všetky unicastové adresy predtým ako budú rozhraniu pridelené, bez ohľadu na to či boli získané bezstavovou konfiguráciou, pomocou DHCPv6 alebo statickou konfiguráciou, prípadné výnimky definuje RFC4862 [17].

Bezstavová autokonfigurácia

Mechanizmus bezstavovej konfigurácie uzlom umožňuje vygenerovať si vlastné adresy za pomoci lokálne dostupných informácii a informácií poskytnutých smerovačmi. Nevyžaduje žiadnu manuálnu podporu, minimálnu podporu od smerovačov a žiadnu spoluprácu s ďalšími servermi. Smerovače poskytujú informácie o prefixoch, ktoré identifikujú podsiete spojené s danou linkou, zatiaľčo uzly si generujú identifikátor rozhrania, ktorý jedinečne identifikuje rozhranie v podsieti. Adresa je vytvorená kombináciou týchto dvoch informácii. Ak by sa v sieti nenachádzali žiadne smerovače, uzol si môže vygenerovať len link-local adresy. Tieto adresy sú však dostatočné pre komunikáciu medzi uzlami pripojenými k rovnakej linke [17].
Bezstavová autokonfigurácia využíva dvojice správ Neighbor Solicitation/Advertisement a Router Solicitation/Advertisement. Obrázok 1.8 zobrazuje výmenu správ medzi uzlom a smerovačom pri bezstavovej autokonfigurácii adresy. Ako prvú zasiela uzol správu RS, v ktorej hľadá dostupné smerovače na danej linke [19].



Obr. 1.8: Výmena ICMPv6 správ medzi uzlom a smerovačom pri bezstavovej autokonfigurácii adresy, prevzaté z [19].

Odpoveďou na túto správu je správa RA, kde smerovač poskytuje prefixy, ktoré sú dostupné na danej linke a podľa ktorých si uzol môže vytvoriť adresu ak majú nastavený príznak pre použitie k bezstavovej konfigurácii, dobu životnosti daných prefixov a základné informácie o zariadení ako napríklad východzí smerovač [20]. Poslednou časťou konfigurácie je vyvolanie správy NS v rámci detekcie duplicitných adries DAD [19].

Metóda EUI-64

Pri vytváraní IPv6 adries uzly v simulátore ns-3 používajú metódu EUI-64 (Extended Unique Identifier) [21]. Táto metóda umožňuje uzlu vytvorenie unikátneho 64 bitového identifikátoru siete na základe 48 bitovej MAC adresy uzlu. Kroky pre vytvorenie identifikátora za pomoci tejto metódy [22] sú:

1. Rozdelenie MAC adresy na dve 24 bitové časti a pridanie 16 bitovej hodnoty hexadecimálne zapísanej ako 0xFFFE medzi tieto dve časti.
2. Invertovanie universal/local príznaku (siedmy bit v poradí) v prvej 24 bitovej časti MAC adresy.

1.3.3 Statické a dynamické smerovanie v IPv6

V druhej časti úlohy (2.2.4) sú ukázané dva typy smerovania v rámci IPv6. Prvým z nich je nastavenie smerovania pomocou statických ciest. V rámci tohto smerovania je ukázaná funkcionalita ICMPv6 správy Redirect, ktorá môže ovplyvniť smerovanie paketov v sieti, ak je to vhodné. Správa Redirect je vyvolaná ak smerovač zistí, že uzol nemá pre danú cieľovú adresu efektívnu next-hop adresu. To je v rámci úlohy docielené neefektívnym nastavením statických ciest. Protokol ICMPv6 túto neefektívnu cestu zistí pri zasielaní ICMPv6 správ o dostupnosti uzlu (Echo Request a

Echo Reply) a správou Redirect ju upraví.
Pri zasielaní ICMPv6 správ o dostupnosti uzlu (Echo Request a Echo Reply) je v rámci úlohy taktiež ukázaná funkcionalita protokolu ICMPv6 pre získanie MAC adresy z danej IP adresy, ktorú v rámci IPv4 vykonáva protokol ARP.
V poslednom bode tejto časti úlohy je do siete pridané dynamické smerovanie pomocou protokolu RIPng (Routing Information Protocol next generation), ktorý je ďalšou verziou protokolu RIP a zahŕňa podporu pre smerovanie v IPv6 sieťach.

Správy Echo Request a Echo Reply

Dvojica správ Echo Request/Echo Reply sa používa pre testovanie dostupnosti cieľového uzla [15].
Každý uzol musí mať implementovanú funkciu ICMPv6 Echo responder, ktorá príjme Echo Požiadavky (ďalej ako Echo Requests) a vytvára korešpondujúce Echo Odpovede (ďalej ako Echo Replies). Uzol by taktiež mal mať na aplikačnej vrstve implementované rozhranie pre vytváranie správ Echo Requests a prijímanie Echo Replies pre účely diagnostiky siete [13].

Echo Request

Správa obsahuje základné časti - Typ, Kód a Kontrolný súčet, ku ktorým sú pridané polia [13]:

- **Identifier - Identifikátor**, ktorý pomáha pri párovaní správy Echo Reply k tejto správe Echo Request. Môže mať aj nulovú hodnotu.
- **Sequence number - Poradové číslo**, ktoré tiež napomáha pri párovaní správy Echo Reply k tejto správe Echo Request. Môže maž aj nulovú hodnotu.
- **Data - Dáta**, obsahujúce nula alebo viac oktetov ľubovoľných dát.

Pre **typ správy Echo Request** je definovaná hodnota **128** a definovanou hodnotou pre kód je len 0. Cieľovou adresou pre túto správu môže byť akákoľvek platná IPv6 adresa. Zdrojovou adresou by mala byť adresa uzlu, ktorý ju odosiela [13].

Echo Reply

Správa obsahuje základné časti - Typ, Kód a Kontrolný súčet, ku ktorým sú pridané polia [13]:

- **Identifier - Identifikátor**, zo správy Echo Request, ktorá túto správu vyvolala.
- **Sequence number - Poradové číslo** zo správy Echo Request, ktorá túto správu vyvolala.
- **Data - Dáta** zo správy Echo Request, ktorá túto správu vyvolala.

Pre **typ správy Echo Reply** je definovaná hodnota **129** a definovanou hodnotou pre kód je len 0. Zdrojová adresa správy Echo Reply, ktorá je odpoveďou na správu Echo Request zaslanú z unicastovej adresy, musí byť rovnaká ako cieľová adresa Echo Request správy. Správa Echo Reply by mala byť odoslaná aj ako odpoveď na Echo Request správu zaslanú na IPv6 multicastovú alebo anycastovú adresu. V tomto prípadne, zdrojovou adresou odpovede musí byť unicastová adresa patriaca rozhraniu, na ktorom bola prijatá správa Echo Request [13].
Dáta prijaté v správe Echo Request musia byť vrátené a nemodifikované v správe Echo Reply. Neexistuje žiadne obmedzenie množstva údajov, ktoré správy Echo Request a Echo Reply môžu obsahovať [13].

ICMPv6 ako ARP pre IPv6

Aby mohli byť cieľovému uzlu doručené správy od zdrojového uzlu, potrebuje zdrojový uzol zistiť MAC adresu cieľového uzlu. Túto funkciu má v rámci IPv4 na starosti protokol ARP. V rámci protokolu IPv6 nová verzia protokolu ARP nie je a túto funkciu má v správe taktiež protokol ICMPv6. Funkcionalitu ARP pre IPv6 má na starosti dvojica správ Neighbor Solicitation/Advertisement, keďže sa jedná o komunikáciu medzi uzlami [23].
O MAC adresu pre daný uzol, či smerovač, si zdrojový uzol žiada zaslaním správy NS - Neighbor Solicitation. Zdrojovou adresou tejto NS správy je link-local adresa zdrojového uzlu a cieľovou adresou je multicastová solicited-node adresa pre IPv6 adresu cieľového uzlu [24].
Odpoveďou na správu NS je správa NA - Neighbor Advertisement. Táto NA správa je zasielaná priamo od uzlu, ktorého MAC adresu bolo žiadané zistiť a jej cieľom je uzol, ktorý o MAC adresu žiadal. A v poli ICMPv6 Option má voľbu Target link-layer address, ktorá obsahuje žiadanú MAC adresu [24].

Solicited-node adresa

Solicited-node adresa je ďalšou multicastovou adresou, ktorú je vyžadované aby si ju uzol na danom rozhraní vytvoril a prímal jej pakety po tom, čo mu je pridelená hocijaká unicastová adresa. Adresa sa vytvára tak, že sa z unicastovej adresy zoberie posledných 24b a pridajú sa k prefixu FF02:0:0:0:0:1:FF00::/104, čím vznikne multicastová solicited-node adresa. To potrebuje uzol urobiť pre všetky svoje unicastové adresy - všetky link-local, ale aj globálne adresy na danom rozhraní [25].

Rozdiel medzi ARP a ICMPv6 NS/NA

Rozdiel spočíva v spôsobe zasielania požiadavky pre zistenie danej MAC adresy. Protokol ARP pri zisťovaní MAC adresy uzla zasiela požiadavku pre túto adresu

každému uzlu v sieti broadcastom - s MAC adresou FF:FF:FF:FF:FF:FF. Pre protokol IPv6 však broadcast definovaný nie je. Používa sa multicast pre ktorého MAC adresy platí, že sú stále z rozsahu 33:33:00:00:00:00 - 33:33:FF:FF:FF:FF. Teda aj multicastová solicited-node adresa bude mať prislúchajúcu MAC adresu s prefixom 33:33:__:__:__:__ doplnenú o posledných 32 bitov z vytvorenej solicited-node adresy [24].
Od každého z uzlov sa vyžaduje prijímanie paketov multicastovej solicited-node adresy pre každú svoju unicastovú adresu. Každý ďalší uzol si vie túto adresu, spolu s jej MAC adresou dopočítať pre akýkoľvek uzol v sieti. Rozdiel od ARP pre zisťovanie MAC adresy od ICMPv6 je práve vo využití multicastu narozdiel od broadcastu [25]. V rámci ICMPv6 nie je požiadavka o MAC adresu odoslaná každému uzlu v sieti, ako v protokole ARP, ale požiadavka príde len na rozhranie uzlu, ktorého sa týka.

Protokol RIPng

Protokol RIPng je IGP (Interior Gateway Protocol) protokol, ktorý využíva Bellman-Ford distace-vector algoritmus k rozhodovaniu o najlepšej ceste k cieľu pomocou počtu skokov ako svoju metriku. Protokol umožňuje uzlom a smerovačom výmenu informácii k definovaniu ciest cez siete využívajúce IP protokol. Na transportnej vrstve RIPng využíva protokol UDP s číslom portu 521. Nevýhody, ktoré tento protokol má sú [26]:

- Najdlhšia cesta sieťou nemôže prekročiť 15 hopov.
- Protokol je náchylný k smerovacím slučkám pri rekonštrukcii smerovacích tabuliek. Špeciálne ak je implementovaný vo veľkých sieťach, ktoré sú tvorené veľkým počtov smerovačov môže protokolu trvať dlhý čas kým slučky vyrieši.
- Pre výber cesty používa len fixnú metriku a nepodporuje výber cesty na základe iných parametrov (napríklad na základe meškania, spoľahlivosti apod.).

Hlavička RIPng paketu obsahuje nasledujúce polia [26]:

- Príkaz (Command) - indikuje, či je v pakete správa pre príkaz o požiadavku alebo pre odpoveď. Správa s príkazom pre požiadavku žiada informáciu o smerovacej tabuľke smerovača. Správy s príkazom pre odpoveď obsahujú okrem polí pre príkaz, verziu aj množinu záznamov pre dané cieľové adresy s ich metrikou a môžu byť zasielané na vyžiadanie alebo periodicky (update správy).
- Číslo verzie (Version number) - špecifikuje verziu RIPng protokolu, ktorú používa smerovač.

Paketu protokolu RIPng sú zasielané na multicastovú adresu skupiny všetkých RIPng smerovačov - ff02::9. Obdobné multicastové adresy má IPv6 definované aj napríklad pre smerovače, ktoré používajú smerovací protokol protokol OSPFv3 - ff02::5, aby si mohli predávať smerovacie informácie a nezaťažovali nimi ostatné uzly [27].

1.3.4 Chybové správy protokolu ICMPv6

V poslednej časti úlohy (2.2.5) pre opis správ protokolu ICMPv6 sú predstavné 3 chybové správy protokolu. Sú to správy Time Exceeded, Destination Unreachable a Packet Too Big.

Správa Destination Unreachable

Táto správa by mala byť vygenerovaná smerovačom alebo IPv6 vrstvou zdrojového uzlu ako odpoveď na paket, ktorý nemôže byť doručený na cieľovú adresu z iných dôvodov ako pre zahltenie siete. Formát správy obsahuje základné časti správy, ktorými sú Typ, Kód a Kontrolný súčet. Ďalej správa obsahuje pole **Nevyužíté (Unused)** a pole, ktoré obsahuje čo **najviac informácii o pakete, ktorý správu vyvolal**, pričom celá správa by nemala prekročiť limit MTU danej siete [13].
Správa Destination unreachable má definovanú má pre svoj **typ** hodnotu **1** a pre jej **kód** sú definované nasledujúce hodnoty [13]:

- **0 - No route to destination**, ak je dôvodom zlyhania doručenia paketu nedostatočný počet záznamov v smerovacej tabuľke uzlu.
- **1 - Communication with destination administratively prohibited**, ak je dôvodom zlyhania doručenia administratívny zákaz (napríklad Firewall filter).
- **2 - Beyond scope of source address**, ak je dôvodom zlyhania doručenia to, že cieľ je mimo dosah zdrojovej adresy. Táto situácia môže nastať len ak rozsah zdrojovej adresy je menší ako dosah cieľovej adresy (napríklad ak má paket ako zdrojovú adresu link-local adresu a globálnu cieľovú adresu) a paket nemôže byť doručený cieľu bez opustenia rozsahu zdrojovej adresy.
- **3 - Address unreachable**, ak sa dôvod zlyhania doručenia nedá priradiť k žiadnemu inému kódu. Nastáva to napríklad pre prípady keď nie je možné získať z cieľovej IPv6 adresy príslušnú linkovú adresy alebo nastal problém so spojením na linkovej vrstve.
- **4 - Port unreachable**, by mal byť uvedený v odpovedi na paket pre ktorý nemá transportný protokol listener, ak daný protokol (napríklad UDP) nemá alternatívny prostriedok k informovaniu odosielateľa.
- **5 - Source address failed ingress/egress policy**, ak je dôvodom zlyhania doručenia, že paket s touto zdrojovou adresou nemá dovolený ingress alebo egress politikou/nariadeniami - ACL.
- **6 - Reject route to destination**, ak je dôvodom zlyhania doručenia to, že cesta k tomu cieľu je trasou, ktorá má byť odmietnutá. To môže nastať ak bol smerovač nakonfigurovaný k odmietnutiu všetkých spojení pre špecifický prefix.

Cieľová adresa tejto správy je skopírovaná z poľa zdrojovej adresy paketu, ktorý ju vyvolal a zdrojovou adresou tejto správy, je adresa smerovača, ktorý ju zasiela [13].

Správa Packet Too Big

Správa Packet Too Big musí byť odoslaná zo smerovača ako odpoveď na paket, ktorý nemôže byť ďalej odoslaný, pretože jeho veľkosť je väčšia ako veľkosť MTU (Maximum Transmission Unit) spojenia, na aké má byť zaslaný [13].
Správa obsahuje základné časti - Typ, Kód a Kontrolný súčet, ku ktorým je pridané pole **MTU**, a podobne ako všetky chybové správy, aj pole **obsahujúce informácie o pakete, ktorý ju vyvolal**. Pre svoj **typ má správa Packet Too Big** definovanú hodnotu **2** a jej kód má definovanú len hodnotu 0, ktorá nenesie žiadnu špecifickú informáciu o chybe. Pole MTU obsahuje hodnotu MTU pre spojenie, na ktoré má byť paket ďalej odoslaný (next-hop link) [13].
Informácia, ktorú poskytuje táto správa je aj súčasťou mechanizmu PMTU (Path MTU Discovery). Myšlienka mechanizmu PMTU spočíva v tom, že zdrojový uzol by mal paket rozdeliť na menšiu alebo rovnakú veľkosť, akú má spojenie s najmenším MTU po ceste, aby boli dáta danému cieľu úspešne doručené. Proces hľadania tejto hodnoty si môže vyžadovať viacero iterácii odoslania paketu a prijatia správy Packet Too Big, keďže po ceste paketu môže byť viacero spojení s nižším MTU, aké si doposiaľ uzol definoval. Implementácia tohto mechanizmu pre uzol nie je povinná. Uzly, ktoré ju nemajú, používajú definovanú minimálnu veľkosť MTU uvedenú v RFC 8200 [28](1280 oktetov) ako maximálnu veľkosť paketu. To vedie v množstve prípadov k používaniu väčšieho množstva malých paketov, aj keď to nie je nutné [29].
Cieľová adresa tejto správy je skopírovaná z poľa zdrojovej adresy paketu, ktorý ju vyvolal a zdrojovou adresou tejto správy, je adresa smerovača, ktorý ju zasiela [13].

Správa Time Exceeded

Ak smerovač prijme paket s nulovou hodnotou v poli Hop limit alebo ak smerovač zníži túto hodnotu na nulu, musí tento paket zahodiť a zaslať správu ICMPv6 Time Exceeded s kódom 0 zdroju tohto paketu. To môže indikovať buď slučku v smerovaní paketu alebo veľmi malú počiatočnú hodnotu v poli Hop limit [13].
Správa obsahuje základné časti - Typ, Kód a Kontrolný súčet ku ktorým je pridané pole **Nevyužité (Unused)** a pole **obsahujúce informácie o pakete, ktorý ju vyvolal**. Pre svoj **typ má správa Time Exceeded** definovanú hodnotu **3** a jej **kód** môže nadobúdať dve hodnoty [13]:

- **0 - Hop limit exceeded in tranzit**, je nastavený v prípade ak paketu klesla hodnota Hop limit na nulu alebo smerovaču prišiel paket s touto hodnotou.

- **1 - Fragment reassembly time exceeded**, sa požíva k oznámeniu vypršania časovaču pre opätovné poskladanie fragmentov (fragment reassembly timeout) špecifikovaný v RFC 8200 [28].

Cieľová adresa tejto správy je skopírovaná z poľa zdrojovej adresy paketu, ktorý ju vyvolal a zdrojovou adresou tejto správy, je adresa smerovača, ktorý ju zasiela [13].

2 Navrhnuté laboratórne úlohy

2.1 Úloha: Konfigurace protokolu BGP

2.1.1 Úvod

Cílem této laboratorní úlohy je přiblížit fungování směrování na úrovni autonomních systémů (AS) za pomoci protokolu BGP.
Autonomní systém (AS) je skupina sítí se společnou směrovací politikou a pod společnou správou. Vlastní autonomní systém mají typicky ISP, datacentra nebo jiné společnosti s rozsáhlou počítačovou sítí. Každý IP rozsah patří do jednoho autonomního systému. Vlastní autonomní systém (a IP rozsah) je ale nezbytný pro tzv. multihoming, tedy připojení sítě k Internetu přes více ISP najednou.
Autonomní systémy vytváří dvouúrovňovou hierarchii směrování na Internetu – dělí směrování na směrování mezi autonomními systémy a na směrování v rámci jednoho autonomního systému. Pro směrování mezi autonomními systémy se používají EGP (Exterior Gateway Protocol) protokoly. Současným standardem je BGP (Border Gateway Protocol) verze 4. Pro směrování v rámci autonomního systému se používá některý z IGP (Interior Gateway Protocols) protokolů – typicky RIP (Routing Information Protocol), OSPF (Open Shortest Path First), IS-IS (Intermediate System to Intermediate System) nebo EIGRP (Enhanced Interior Gateway Protocol). Směrování v rámci AS je interní záležitostí AS a rozhoduje o něm správce AS – nemá vliv na část internetu vně AS.
Aby BGP mohl nalézt cestu do určitého AS, musí mít každý AS přiděleno jednoznačné číslo, tzv. Autonomus System Number (ASN). Číslo autonomního systému se ukládá do 16bitového pole, takže muže nabývat hodnoty z rozsahu 0 - 65535.
Směrování mezi autonomními systémy má charakteristické požadavky, které se nevyskytují v interním směrování. Směrovací tabulky velké množství záznamů, nejdůležitějším kritériem nebývá vzdálenost, ale posuzují se nastavitelné parametry zohledňující například cenu a dodatečná pravidla aplikovaná v závislosti na zdroji, cíli, seznamu tranzitních autonomních systémů a dalších atributech. Vzhledem k velkému počtu záznamů se v případě změn v topologii vyměňují pouze informace o změnách, nikoliv celé směrovací tabulky jako je tomu v případě protokolu RIP.

Popis úlohy

Cílem této laboratorní úlohy je přiblížit fungování směrování na úrovni autonomních systémů (AS) za pomoci protokolu BGP.

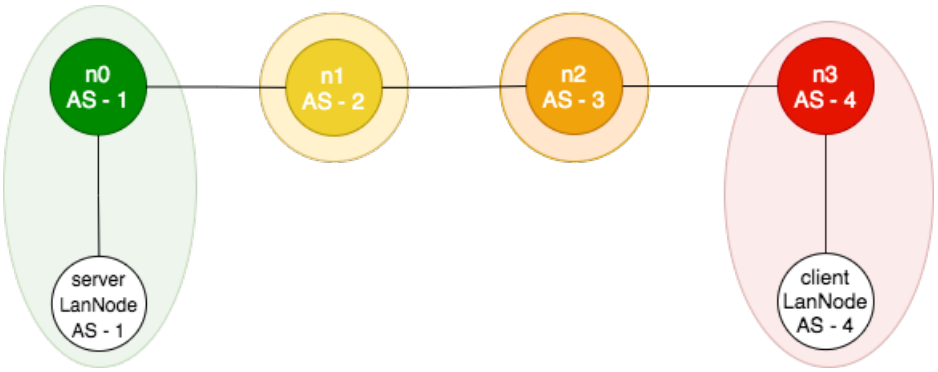
Laboratorní úloha popisuje:

1. jak se vytváří sousedství mezi AS
2. přidání dalšího AS mezi již existující spojení systémů
3. pakety, ve kterých jsou přenášeny zprávy protokolu BGP
4. směrovací tabulky směrovačů po konfiguraci jednotlivých sousedství
5. simulování výpadku spojení v topologii
6. změnu trasy provozu pomocí přiřazení PeerLinku mezi směrovači

Samostatním úkolem je přidání dalšího AS a jeho konfigurace pomocí protokolu BGP, tak aby správně spolupracoval s již vytvořenými AS.

Výchozí topologie

Obrázek 2.1 znázorňuje výchozí topologii pro tento úkol. Topologie se skládá ze čtyř autonomních systémů, z nichž každý má přiděleno své ASN číslo - *AS-1, AS-2, AS-3, AS-4*.
Tato topologie je zjednodušená pro účely tohoto úkolu. V reálné implementaci může být směrovačů v daných AS i více. Jednotlivé `nX` uzly představují výchozí směrovače daného AS. K AS 1 je přidán uzel `serverLanNode`, který představuje lokální síť 10.1.1.0/24 v AS - 1. Této síti byl adresní rozsah přidělen výchozím směrovačem daného AS směrovačem - `n0`. Podobně je to pro AS - 4, kde uzel `clientLanNode` reprezentuje lokální síti 10.1.3.0/24. O směrování v této síti se již nestará protokol eBGP, ale zvolený IGP protokol (RIP, OSPF, IBGP apod.).



Obr. 2.1: Výchozí topologie

Vytvoření výchozí topologie

Na začátek je třeba se přemístit v terminálu do složky example, kde se nachází předpřipravený soubor `bgp_start.cc`.

```
cd /home/student/dce/source/ns-3-dce/myscripts/
    ns-3-dce-quagga/example/
```

Soubor otevřete v editoru Sublime jako root příkazem:

```
sudo subl bgp_start.cc
```

V tomto souboru je definována výchozí topologie jako je zobrazena na obrázku 2.1. V následujících částech je soubor postupně popsán. V první částí úlohy, po seznámení se s předdefinovaným kódem, budete nastavovat sousedství mezi směrovači *n2* a *n3* podle již nastavených příkladů. Místo pro doplnění kódu je vyznačeno komentářem *//VYCHOZI TOPOLOGIE: Zde dopiste vytvoreni sousedstvi mezi smerovaci n2 a n3 (sit 10.0.3.0/30).*

Definice modulů a parametrů

Na začátku souboru `bgp_start.cc` jsou definovány moduly, které budou potřebné pro tento úkol.

```
#include "ns3/network-module.h"
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/dce-module.h"
#include "ns3/quagga-helper.h"
#include "ns3/point-to-point-helper.h"
#include "ns3/netanim-module.h"
#include "ns3/ipv4-dce-routing-helper.h"
#include "ns3/constant-position-mobility-model.h"
#include "ns3/applications-module.h"
#include "ns3/gnuplot.h"
#include "ns3/csma-module.h"
#include "ns3/mobility-module.h"
```

Dále je specifikovný obor názvů (namespace) pro používané typy, proměnné a funkce. Pro tento úkol je to namespace NS3, protože pro simulování dané topologie bude použit NS3 simulátor.
Na dalším řádku je deklarována a inicializována globální proměnná - `endTime`. Hodnota proměnné `endTime` udává čas trvání simulace. Poslední řádek definuje název komponenty, pod kterou budou zobrazované logovací zprávy daného programu.

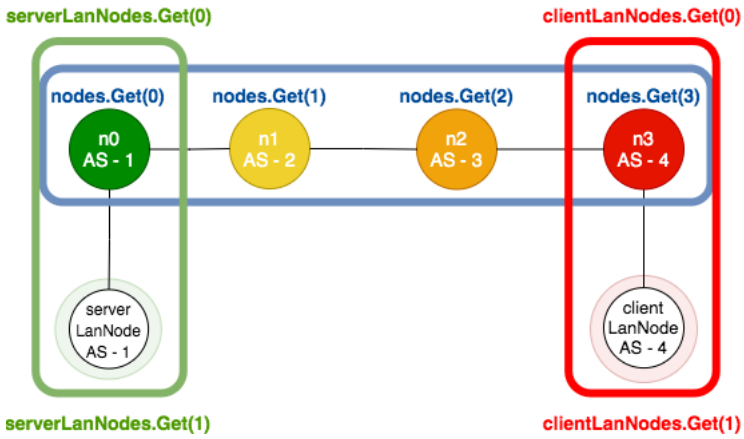
```
using namespace ns3;
double endTime = 300.0;
NS_LOG_COMPONENT_DEFINE ("BGP_routing");
```

Definice uzlů

Prostředí NS3 vytváří všechny použité komponenty sítě jako uzly. Vytvoření uzlů je uvedené již ve funkci `main`. Komentáře *//PRIDANI AS-5:*, *//PRIDANI AS-6:*, *//VYPADEK:* a *//PEERLINK:* zatím ignorujte, označují místo na dopnění nebo změnu kódu následujících úkolů a budete je později postupně doplňovat.

```
NodeContainer nodes;
nodes.Create (4);

NodeContainer serverLanNodes;
serverLanNodes.Add(nodes.Get(0));
serverLanNodes.Create (1);

NodeContainer clientLanNodes;
clientLanNodes.Add(nodes.Get(3));
clientLanNodes.Create(1);
```

Tyto uzly budou v topologii představovat směrovače. K vytvoření výchozí topologie budeme potřebovat 3 skupiny uzlů/směrovačů - hraniční směrovače pro každý AS, směrovače v lokální síti 10.1.1.0/24 a směrovače v lokální síti 10.1.3.0/24. Tyto směrovače si, pro lepší práci s nimi, vložíme do proměnných typu `NodeContainer`, z nichž každá proměnná bude reprezentovat jednu skupinu.
První skupinu směrovačů bude reprezentovat proměnná `nodes`. V této skupině jsou pro výchozí topologii vytvořeni 4 směrovače. Tedy tolik, kolik autonomních systémů vyžadujeme.
Druhá skupina směrovačů (lokální síť 10.1.1.0/24) je reprezentována proměnnou `serverLanNodes`. Jako první je do této skupiny přidán uzel `n0`, který již není nutné vytvářet, stačí ho pouze přiřadit i k druhé skupině a to voláním metody `serverLanNodes.Add (nodes.Get (0))`. Tato skupina má obsahovat ještě jeden uzel - `serverLanNode` a ten je vytvořen obdobně jako v předchozí skupině.
Třetí skupinou směrovačů je lokální síť 10.1.3.0/24, která je reprezentována proměnnou `clientLanNodes`. Do této skupiny patří směrovač `n3` a je v ní třeba taky vytvořit jeden další směrovač - `clientLanNode`.
Při přiřazování směrovačů je třeba dávat si pozor na indexy daných prvků. Je třeba dbat na to, že indexování prvků začíná na čísle 0. Tedy směrovač `n0` můžeme získat dvěma způsoby, a to buď z proměnné `nodes.Get (0)` nebo `serverLanNodes.Get (0)`.



Obr. 2.2: Rozdělení uzlů do skupin

Vytvoření přenosových kanálu - L2 vrstva

V tomto úkolu jsou použity přenosové kanály typu - `PointToPoint`, které vytvářejí spojení směrovačů. Kanál typu `PointToPoint` představuje spojení bod-bod, a slouží tak k propojení jenom dvou uzlů mezi sebou. Pro spojení je možné definovat vlastnosti jako rychlost přenosu dat či zpoždění.

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate",
                                 StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay",
                                  StringValue ("2ms"));
```

K vytvoření propojení směrovačů slouží pomocná třída `PointToPointHelper`. Toto propojení je uloženo v proměnné typu `NetDeviceContainer`. Pro příklad, proměnná `n0server1` reprezentuje spoj směrovače `n0` a směrovače `serverLanNode`.

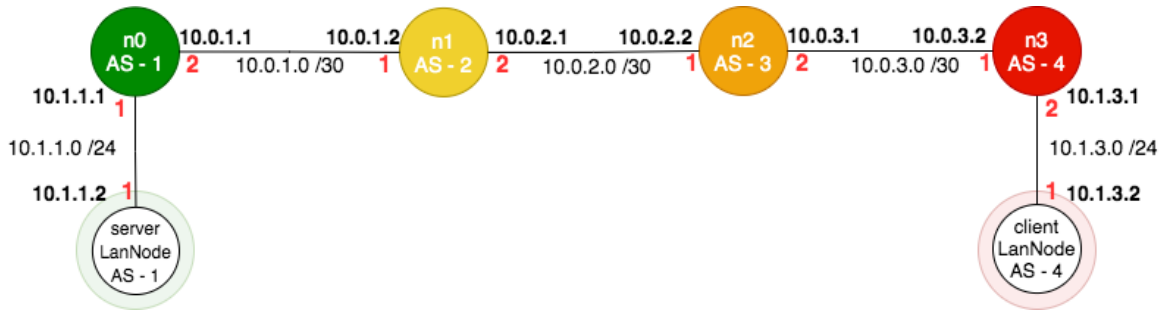
```
NetDeviceContainer n0server1 = pointToPoint.Install(
nodes.Get (0), serverLanNodes.Get (1));
NetDeviceContainer n0n1 = pointToPoint.Install(
nodes.Get (0), nodes.Get (1));
NetDeviceContainer n1n2 = pointToPoint.Install(
nodes.Get (1), nodes.Get (2));
NetDeviceContainer n2n3 = pointToPoint.Install(
nodes.Get (2), nodes.Get (3));
NetDeviceContainer n3client1 = pointToPoint.Install (
nodes.Get (3), clientLanNodes.Get(1));
```

Aby jednotlivé uzly mohly mezi sebou komunikovat, je třeba na každý z nich nainstalovat TCP/IP stack pomocí třídy `InternetStackHelper` a přiřadit jim adresy

pomocí třídy `Ipv4AddressHelper`.

```
Ipv4AddressHelper ipv4AddrHelper;
InternetStackHelper stack;
Ipv4DceRoutingHelper ipv4RoutingHelper;
stack.SetRoutingHelper(ipv4RoutingHelper);
stack.Install(nodes);
stack.Install(serverLanNodes.Get(1));
stack.Install(clientLanNodes.Get(1));
```

Podle obrázku 2.3 jsou dále jednotlivým směrovačům přidělované IPv4 adresy podle sítě, ve které se nacházejí. Jako první je pomocí metody `Ipv4AddressHelper ::SetBase()` definovaný adresný rozsah, ze kterého budou v dalším kroku přidělené adresy jednotlivým uzlům v daném spoji.
Adresy, které byly daným spojem přiřazeny budou definovány v proměnné typu `Ipv4InterfaceContainer`. Tato proměnná bude obsahovat adresy rozhraní jednotlivých směrovačů, přičemž první přidělenou adresou z definovaného rozsahu je adresa s nejnižším možným číslem. Pro adresný rozsah 10.0.1.0 s maskou 255.255.255.252 to je adresa 10.0.1.1, která bude přiražena směrovači `n0`.



Obr. 2.3: Výchozí topologie s přidělenými adresami (označené černě) pro jednotlivá rozhraní (označené červeně).

Je dobré dbát na pořadí jak byly smerovače přidávány do jednotlivých spojů, protože podle tohoto pořadí jim budou následně přidělovány adresy. Například směrovači `n0` bude přiražena první možná adresa z adresního rozsahu 10.1.1.0/24 a také z adresního rozsahu 10.0.1.0/30, jelikož do obou spojů `n0server1` a `n0n1` byl přidán jako první, přičemž adresu 10.1.1.1 bude mít první rozhraní směrovače `n0` a adresu 10.0.1.1 jeho druhé rozhraní. Na obrázku 2.3 jsou červeně zapsány čísla rozhraní daného směrovače a černě adresa, která danému rozhraní náleží.

```
ipv4AddrHelper.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i0iserver1 =
    ipv4AddrHelper.Assign(n0server1);
```

```
ipv4AddrHelper.SetBase ("10.0.1.0", "255.255.255.252");
Ipv4InterfaceContainer i0i1 = ipv4AddrHelper.Assign (n0n1);

ipv4AddrHelper.SetBase ("10.0.2.0", "255.255.255.252");
Ipv4InterfaceContainer i1i2 = ipv4AddrHelper.Assign (n1n2);

ipv4AddrHelper.SetBase ("10.0.3.0", "255.255.255.252");
Ipv4InterfaceContainer i2i3 = ipv4AddrHelper.Assign (n2n3);

ipv4AddrHelper.SetBase ("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer i3iclient1 =
    ipv4AddrHelper.Assign(n3client1);
```

Protože má komunikace procházet z uzlu `clientLanNode` na uzel `serverLanNode`, jsou v rámci zjednodušení úlohy v kódu vytvořené statické cesty pro tyto uzly, aby v rámci své lokální sítě věděli, kam mají posílat provoz mimo jejich síť. V reálné síti by se měl o toto směrování, namísto statických cest, postarat vybraný IGP protokol dané sítě, například RIP či OSPF.

```
Ipv4StaticRoutingHelper staticHelper;
Ptr<Ipv4> ipv4 = serverLanNodes.Get(1)->GetObject<Ipv4>();
Ptr<Ipv4StaticRouting> Ipv4stat =
staticHelper.GetStaticRouting(ipv4);
Ipv4stat->SetDefaultRoute(i0iserver1.GetAddress(0,0), 1, 0);

Ptr<Ipv4> ipv4_1 = clientLanNodes.Get(1)->GetObject<Ipv4>();
Ptr<Ipv4StaticRouting> Ipv4stat_1 =
staticHelper.GetStaticRouting(ipv4_1);
Ipv4stat_1->SetDefaultRoute(i3iclient1.GetAddress(0), 1, 0);
```

Konfigurace protokolu BGP

V následující části bude popsána konfigurace protokolu BGP pro vytvořené směrovače. Jelikož simulátor NS3 sám o sobě neobsahuje podporu pro konfiguraci protokolu BGP je k tomuto účelu použita nástavba s názvem Quagga. Quagga spolupracuje s jádrem operačního systému a vytváří API pro možnost používání různých síťových nástrojů a protokolů, které v simulátoru NS3 nejsou standardně implementovány. Toto API je reprezentováno nainstalovaným dalšího stacku pro dané uzly. K tomuto účelu slouží třída `DceManagerHelper`.

```
DceManagerHelper processManager;
```

```
processManager.SetNetworkStack ("ns3::Ns3SocketFdFactory");
processManager.Install (nodes);
processManager.Install (serverLanNodes.Get(1));
processManager.Install (clientLanNodes.Get(1));
```

Pro samotné nastavení protokolu BGP je použita třída `QuaggaHelper` pomocí které na jednotlivých směrovačích nakonfigurujeme protokol BGP. Jak je zobrazeno na obrázku 2.2, každý ze směrovačů v proměnné `nodes` reprezentuje hraničný směrovač pro jeden AS.
Pro výměnu směrovacích informací mezi těmito směrovači jsou v protokolu BGP definována tzv. sousedství. Sousedství se vytváří mezi dvěma směrovači, které jsou hraničními směrovači sousedních AS. Toto sousedství se vytváří pomocí funkce `quagga::BgpAddNeighbor(směrovač=uzel, IP adresa souseda, číslo AS souseda)`, jejíž parametry jsou:

- směrovač, pro který bude dané sousedství vytvořeno
- IP adresa sousedního směrovače
- číslo AS sousedního směrovače.

Na příslušném místě kódu označeném komentářem *//VYCHOZÍ TOPOLOGIE: Zde dopište vytvoření sousedství mezi směrovači* ***n2*** *a* ***n3*** *(síť 10.0.3.0/30)* teď doplňte vytvoření sousedství mezi směrovači `n2` a `n3` dle vzoru již vytvořených sousedství se správnými IP adresami dle obr. 2.3.

```
QuaggaHelper quagga;
quagga.EnableBgp (nodes);

// n0-n1
quagga.BgpAddNeighbor(nodes.Get(0), "10.0.1.2",
                      quagga.GetAsn (nodes.Get (1)));
quagga.BgpAddNeighbor(nodes.Get(1), "10.0.1.1",
                      quagga.GetAsn (nodes.Get (0)));
// n1-n2
quagga.BgpAddNeighbor(nodes.Get(1), "10.0.2.2",
                      quagga.GetAsn (nodes.Get (2)));
quagga.BgpAddNeighbor(nodes.Get(2), "10.0.2.1",
                      quagga.GetAsn (nodes.Get (1)));

//VYCHOZI TOPOLOGIE: Zde dopiste vytvoreni sousedstvi mezi
    smerovacmi n2 a n3 (sit 10.0.3.0/30)
//n2-n3

quagga.EnableZebraDebug (nodes);
```

```
quagga.Install (nodes);
quagga.Install (clientLanNodes.Get(1));
```

Definice UDP aplikace

K ověření zda mohou dané směrovače mezi sebou komunikovat a zda je tak nastavení směrování v síti v pořádku je vytvořena aplikace `UdpEchoServer/Client`. Tato aplikace se skládá ze dvou částí - *serveru* a *klienta*, kde klient zasílá serveru UDP pakety a ten na ně odpovídá.
Jako první je zvoleno, na jakém směrovači má běžet UDP server. V případě definované topologie to je směrovač `serverLanNode` s adresou 10.1.1.2. Pro danou aplikaci potřebujeme nastavit čas, kdy bude na daném uzlu spuštěna a kdy má být ukončena.

```
UdpEchoServerHelper echoServer (7);
ApplicationContainer serverApps =
               echoServer.Install(serverLanNodes.Get(1);
serverApps.Start (Seconds (20.0));
serverApps.Stop (Seconds (endTime));
```

Dále je třeba definovat jaký směrovač bude UDP klientem. Pro klienta je třeba specifikovat adresu UDP serveru, kam bude pakety posílat, kolik paketů, v jakém intervalu jich má zasílat a jakou budou mít velikost. Klientem bude uzel `clientLanNode` s adresou 10.1.3.2. Tedy UDP komunikace bude reprezentovat propojení lokálních sítí.

```
UdpEchoClientHelper echoClient (i0iserver1.GetAddress(1), 7);
echoClient.SetAttribute ("MaxPackets", UintegerValue(5));
echoClient.SetAttribute ("Interval", TimeValue(Seconds(1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps =
            echoClient.Install(clientLanNodes.Get(1));
clientApps.Start(Seconds (21.0));
clientApps.Stop(Seconds (endTime));
```

Definice výstupů

Pro získání trasovacích souborů, které budou obsahovat pakety procházející nadefinovanou sítí slouží metoda `PointToPointHelper::EnablePcapAll("filename")`. Metoda vytvoří soubor typu PCAP pro každé rozhraní směrovače v proměnné `pointToPoint`. Například pro směrovač `n1` budou vytvořeny 2 pcap soubory - `p2p_nodes-1-0.pcap` pro rozhraní číslo 1 s adresou 10.0.1.2 a `p2p_nodes-1-1.pcap`

pro rozhraní číslo 2 s adresou 10.0.2.1. Čísla rozhraní směrovačů spolu s jejich adresami jsou popsány na obrázku 2.3 nebo v tabulce 2.1.

Směrovač - č.AS	**Rozhraní**	**IP adresa**	**PCAP**
n0	1	10.1.1.1	p2p_nodes_0-0.pcap
AS - 1	2	10.0.1.1	p2p_nodes_0-1.pcap
n1	1	10.0.1.2	p2p_nodes_1-0.pcap
AS - 2	2	10.0.2.1	p2p_nodes_1-1.pcap
n2	1	10.0.2.2	p2p_nodes_2-0.pcap
AS - 3	2	10.0.3.1	p2p_nodes_2-1.pcap
n3	1	10.0.3.2	p2p_nodes_3-0.pcap
AS - 4	2	10.1.3.1	p2p_nodes_3-1.pcap
serverLanNode - AS-1	1	10.1.1.2	p2p_nodes_4-0.pcap
clientLanNode - AS-4	1	10.1.3.2	p2p_nodes_5-0.pcap

Tab. 2.1: Tabulka s výpisem adres a PCAP souborů pro rozhraní směrovačů

Pro získání záznamů směrovacích tabulek ze směrovačů slouží metoda `Ipv4DceRoutingHelper::PrintRoutingTableEvery()`, ve které parametry specifikují jak často se má získat záznam směrovací tabulky daného směrovače, z jakého směrovače směrovací tabulku chceme získat a kam se má daná tabulka zapsat.

```
pointToPoint.EnablePcapAll ("bgp_start/p2p_nodes");

Ptr<OutputStreamWrapper> routingTable_server =
    Create<OutputStreamWrapper>(
        "bgp_start/bgp-RT_server.log", std::ios::out);
ipv4RoutingHelper.PrintRoutingTableEvery(
    Seconds(50), serverLanNodes.Get(1), routingTable_server);

Ptr<OutputStreamWrapper> routingTable_n0 =
Create<OutputStreamWrapper>("bgp_start/bgp-RT_n0.log",
                                  std::ios::out);
ipv4RoutingHelper.PrintRoutingTableEvery(
    Seconds(50), nodes.Get(0), routingTable_n0);
                                      ...
```

Dále je možné definovanou topologii vykreslit pomocí aplikace NetAnim, která vytvoří XML soubor s daty o provozu. Daný soubor pak bude spustitelný pomocí aplikace NetAnim.

```
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
```

```
                                    "MinX", DoubleValue (10),
                                    "MinY", DoubleValue (10),
                                    ... );
    mobility.SetMobilityModel
    ("ns3::ConstantPositionMobilityModel");
    mobility.Install (nodes);
    mobility.Install (serverLanNodes.Get(1));
    mobility.Install (clientLanNodes.Get(1));

AnimationInterface anim("bgp_animation.xml");
anim.SetConstantPosition(serverLanNodes.Get(1), 1.0, 8.0, 0);
anim.UpdateNodeDescription(serverLanNodes.Get(1), "Server");
anim.UpdateNodeColor(serverLanNodes.Get(1), 50, 205, 50);
...
```

Posledním krokem v kódu je nastavení času ukončení simulace a její spuštění a ukončení.

```
Simulator::Stop (Seconds (endTime));
Simulator::Run ();
Simulator::Destroy ();
```

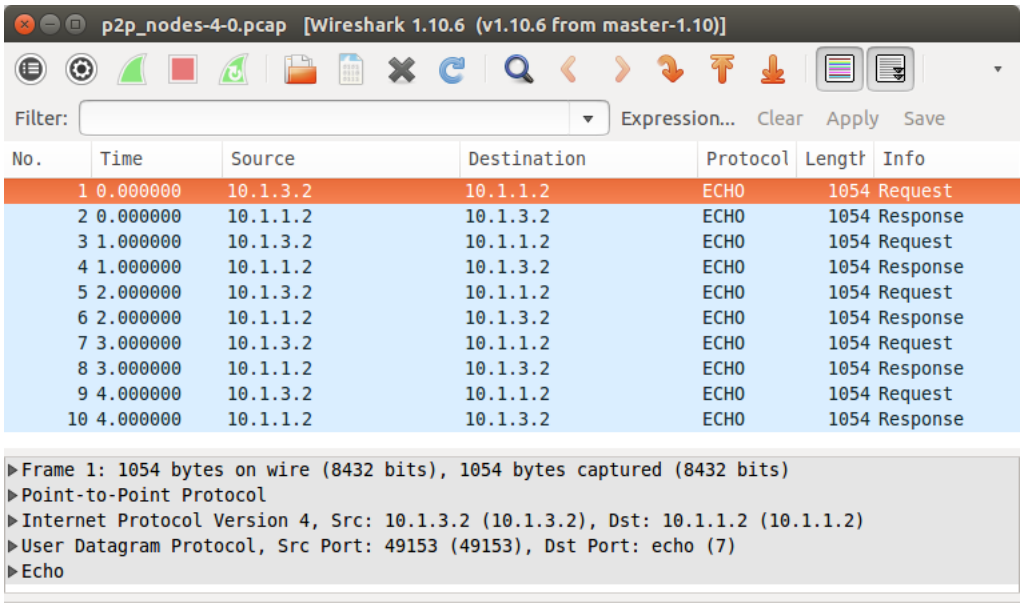
2.1.2 Spuštění a analýza výstupů

Takto vytvořenou síť spustíme příkazem z terminálu:

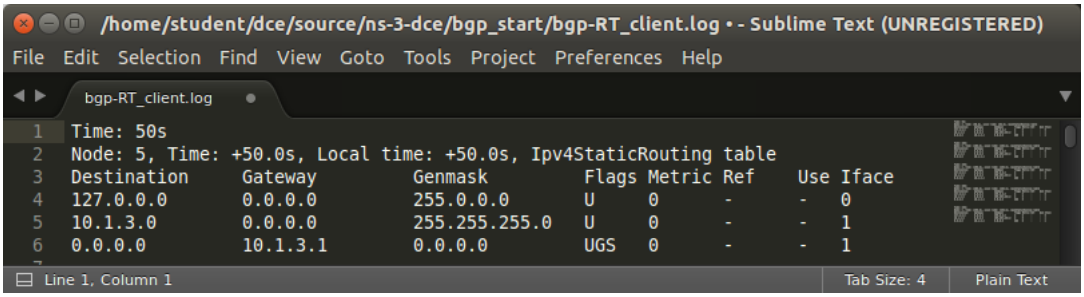
```
cd /home/student/dce/source/ns-3-dce/myscripts/ns-3-dce-quagga/
sudo ./waf --run bgp_start
```

Vytvořené PCAP soubory a směrovací tabulky je možné najít ve složce `/home/student/dce/source/ns-3-dce/bgp_start/`. Pokud veškerá konfigurace proběhla správně, soubor `p2p_nodes-4-0.pcap` obsahuje pakety pro uzel `serverLanNode` a měl by obsahovat pouze pakety UDP protokolu, které směrovač přijal jako UDP server. Obsah souboru by se měl shodovat s obrázkem 2.4. Z těchto paketů je vidět, že UDP server na tyto zprávy zpět klientovi odpověděl na adresu 10.1.3.2. To si můžeme ověřit v souboru `p2p_nodes-5-0.pcap` pro uzel `clientLanNode`, který by měl rovněž obsahovat pouze pakety protokolu UDP.

Dále si najděte směrovací tabulku uzlu `clientLanNode` (klienta) (soubor `bgp-RT_client.log`), ta by se měla shodovat s obrázkem 2.5. I přesto, že tato tabulka neobsahuje záznam o tom, jak se má k síti 10.1.1.0/24 dostat, ví že má veškerý zbylý provoz *0.0.0.0* směřovat na výchozí směrovač daného AS - 10.1.3.1, který směrovací informaci k adrese serveru získal protokolem BGP.

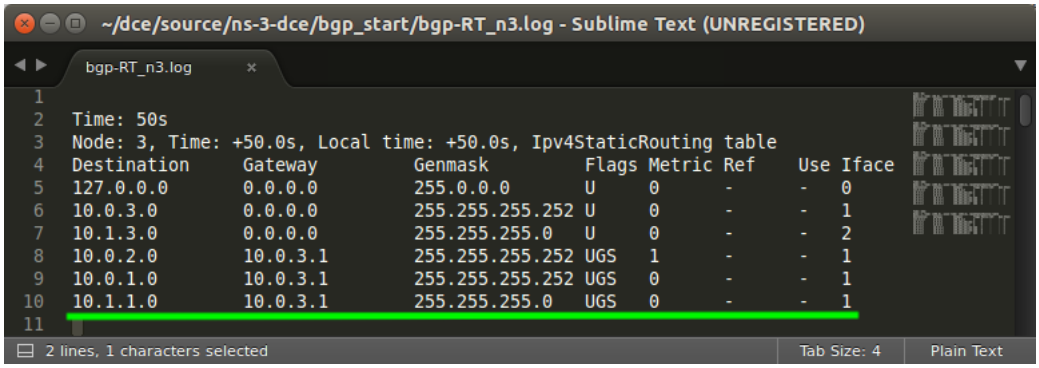


Obr. 2.4: UDP pakety směrovače serverLanNode



Obr. 2.5: Směrovací tabulka směrovače clientLanNode

Výchozím směrovačem pro AS - 4 je `n3`. Směrovací tabulku směrovače `n3` obsahuje soubor `bgp-RT_n3.log` obrázek 2.6, kde je vidět, že provoz pro síť 10.1.1.0/24 je dále posílán přes rozhraní číslo 1.

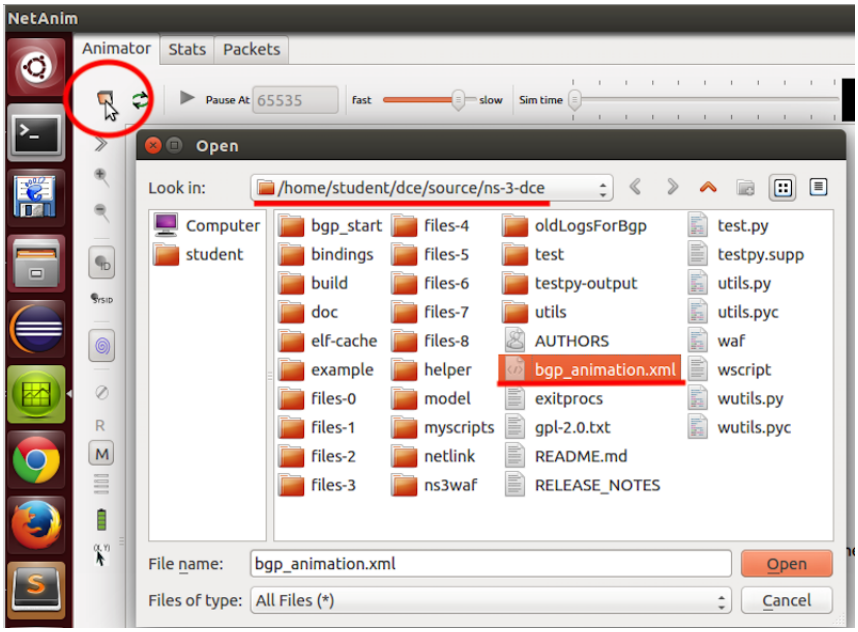


Obr. 2.6: Směrovací tabulka směrovače n3

Pro vizualizaci provozu BGP a UDP zpráv mezi směrovači si spusťte animaci pomocí aplikace NetAnim. Klikněte na ikonku aplikace 2.7.

Obr. 2.7: Ikona aplikace NetAnim

Dále klikněte na vyhledání a otevření souboru v levém horním rohu aplikace 2.8.



Obr. 2.8: Otevření souboru v NetAnim aplikaci

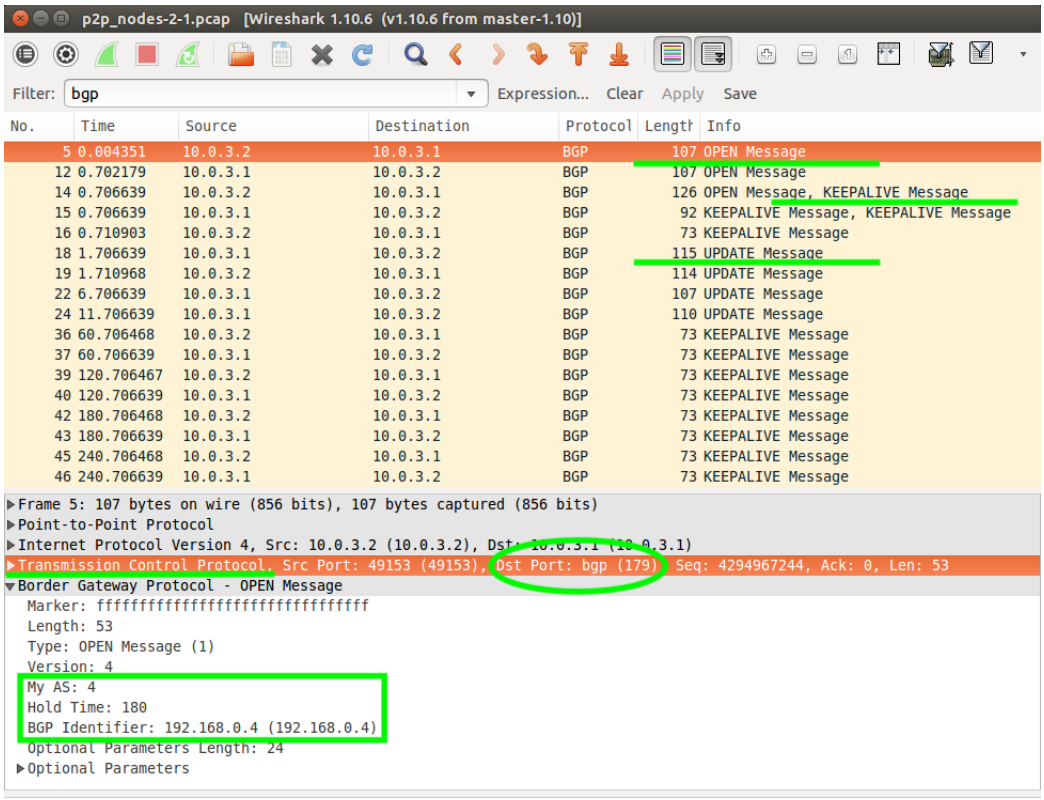
Soubor *bgp_animation.xml* se nachází ve složce */dce/source/ns-3-dce*. Zobrazená topologie by měla shodovat s obrázkem 2.9.

Obr. 2.9: Výchozí topologie v NetAnim

Po spuštění animace je vidět chod jednotlivých paketů při sjednávání BGP sousedství směrovačů jakož i chod UDP paketů. Když komunikace proběhla úspěšně je možné předpokládat, že směrování mezi AS pomocí protokolu BGP bylo nastaveno správně.

Analýza BGP sousedství

Průběh vytvoření a fungování sousedství mezi AS je možné vidět v trasovacím souboru pro jakýkoliv z hraničních směrovačů AS. Při správném nastavení sousedství mezi směrovači `n2` a `n3`, by měl obsah souboru `p2p_nodes-2-1.pcap`, po vyfiltrování protokolu BGP, odpovídat obrázku 2.10.



Obr. 2.10: BGP zprávy směrovače n2 na rozhraní číslo 2

V něm je možné vidět 3 typy zpráv protokolu BGP - OPEN, KEEEPALIVE a UPDATE. Na obrázku 2.10 je z paketu také vidět jaký protokol (TCP) a číslo portu (179) BGP používá na transportní vrstvě. Zprávou typu OPEN začíná vytvoření sousedství mezi směrovačem `n3` a `n2`.

Hlavními parametry zprávy OPEN jsou:

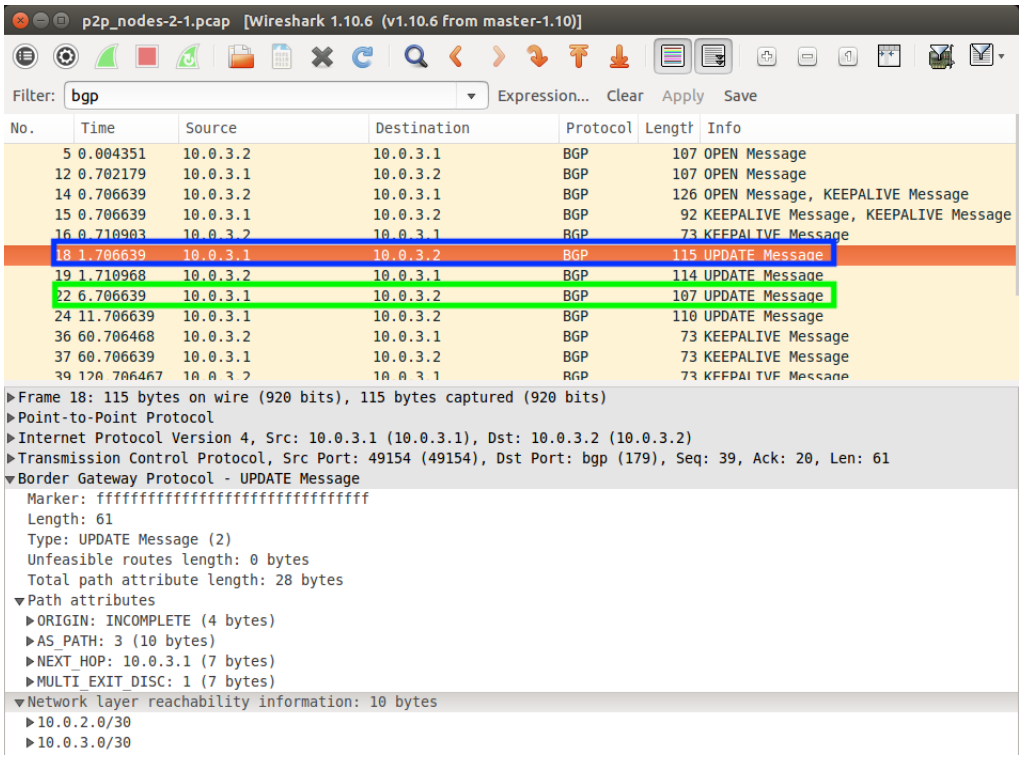
- *My AS*, ve kterém směrovač posílá sousedovi informaci o čísle svého AS,
- *Hold Time*, ve kterém je definována doba po jakou si má soused vytvoření tohoto spojení zapamatovat jako aktivní

- *BGP Identifier*, který směrovači přidělil BGP protokol. Tento identifikátor je důležitý pokud se v rámci jednoho AS nachází několik hraničních směrovačů.

Pro úspěšné vytvoření sousedství musí být zpráva OPEN potvrzena zprávou KEEPALIVE.

Časovač *Hold Time* se po vytvoření sousedství a pak po dalším obdržení zprávy typu KEEPALIVE nebo UPDATE vynuluje. Tedy pokud by směrovač `n2` nedostal zprávu typu KEEPALIVE nebo UPDATE od směrovače `n3` do 180 sekund, toto sousedství by považoval za neplatné.

Pro směrování je velmi důležitá zpráva typu UPDATE. V této zprávě posílá sousední směrovač informaci o tom, k jakým sítím zná cestu. Tedy pokud si sousední směrovač přidá nějakou informaci do své směrovací tabulky, poskytne informaci o této síti i svému sousedovi.



Obr. 2.11: BGP zprávy typu UPDATE na směrovači n2 rozhraní číslo 2

Na obrázku 2.11 je možné vidět, že směrovač `n2` (10.0.3.1) poslal směrovači `n3` (10.0.3.2) po vytvoření sousedství alespoň 2 zprávy typu UPDATE.

V této zprávě protokol BGP posílá všechny důležité údaje o směrování. V poli Path Attributes jsou to:

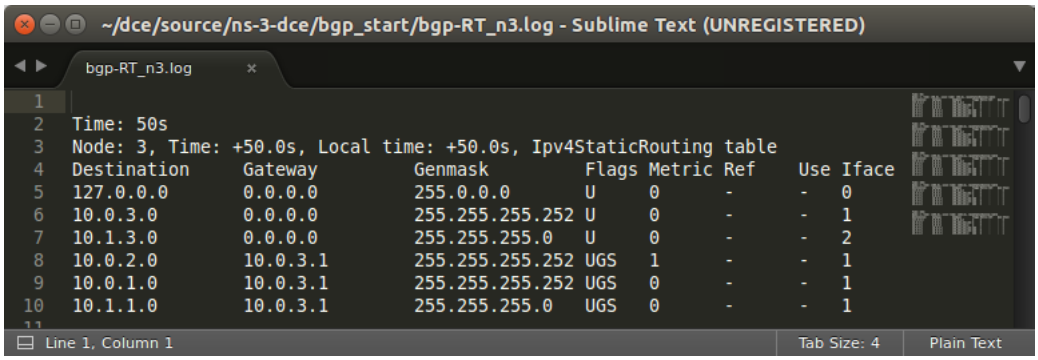
- *ORIGIN*, který definuje původce směrovací informace, obvykle je to IGP protokol, BGP protokol nebo jiný způsob zjištění směrovací informace

- *AS_PATH* je seznam AS, kterými zpráva prošla a tedy popisuje cestu k cílové síti uvedené v poli NLRI - Network Layer Reachability Information
- *NEXT-HOP* je IP adresa dalšího skoku, který by měl být použit pro směrování do cílové sítě, která je uvedená v části NLRI - Network Layer Reachability Information.
- *MULTI_EXIT_DISC* nebo *LOACL_PREF*, které mohou ovlivnit směrování provozu.

V první UPDATE zprávě od směrovače `n2 - src:10.0.3.1` (vyznačena modře), jak je vidět v části paketu *Network Layer Reachability Information*, byly informace o dostupnosti sítí 10.0.2.0/30 a 10.0.3.0/30, které má směrovač `n2` připojené přímo. Ve druhé a třetí UPDATE zprávě od směrovače `n2 - Source: 10.0.3.1` (vyznačena zelenou barvou) je vidět, že směrovač navázal sousedství se směrovačem `n1`, který mu poskytl informaci o sítích 10.0.1.0/30 a 10.1.1.0/24, a po zapsání si této informace do své směrovací tabulky, ji podává i svému sousedovi - směrovači `n3`.

Analýza směrovacích tabulek

Směrovací tabulka směrovače obsahuje informace o sítích, které získal protokolem BGP. Na obrázku 2.12 je směrovací tabulka směrovače `n3`. Podstatným parametrem je sloupec s názvem *Iface*, který specifikuje rozhraní na směrovači, kterým bude paket do cílové sítě odeslán. Čísla těchto rozhraní jsou pro lepší orientaci napsané i na obrázku 2.3 červenou barvou nebo v tabulce 2.1.



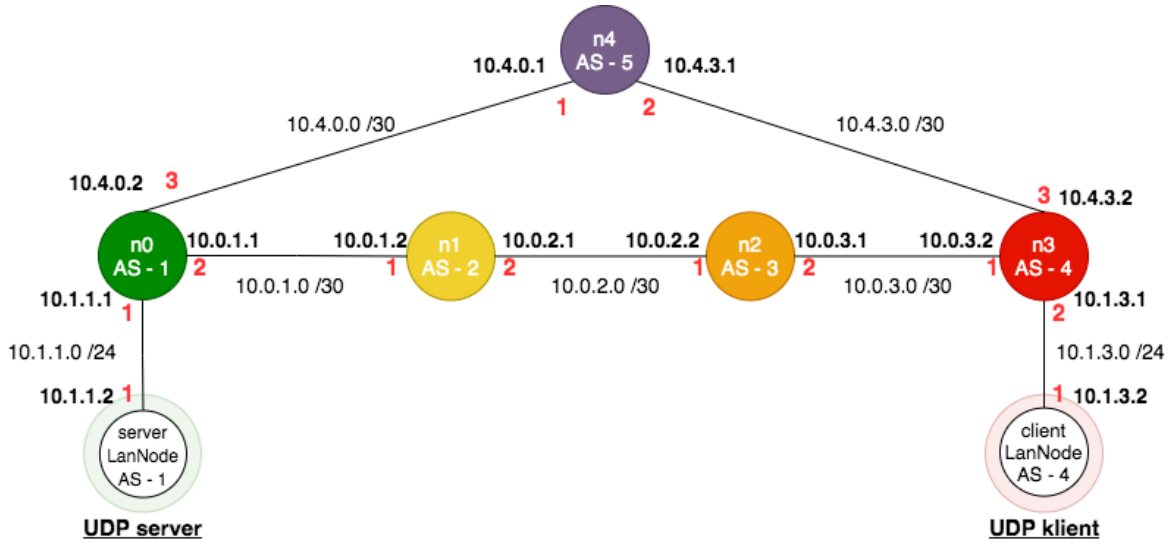
Obr. 2.12: Směrovací tabulka směrovače n3

Směrovací tabulky by měly udávat i informaci o tom, jakým protokolem se o dané síti dozvěděli. Jelikož simulátor NS3 nemá přímo podporu pro protokol BGP, tento detail při výpisu směrovacích tabulek nezobrazuje. Nicméně jistým způsobem tuto informaci reprezentuje sloupec Flags, kde značka U vyjadřuje, že dané spojení je aktivní, značka G, že síť není připojena přímo, ale jedná se o předání na Gateway

a značka S vyjadřuje to, že informace o této síti byla na směrovač přidána staticky příkazem `route`. Co je v tomto případě pravda, protože takto quagga prodává informaci o tom, že tato síť je dostupná přes protokol BGP.

2.1.3 Přidání nového AS

V této části úlohy bude ukázáno jak přidat a nakonfigurovat další AS do základní topologie. Přidán bude AS s číslem 5, který bude reprezentován směrovačem `n4` a bude propojen se směrovačem `n0` s AS-1 a směrovačem `n3` s AS-4. Nová topologie je zobrazena na obrázku 2.13. Do kódu postupne doplňte príslušné časti označené komentářem *//PRIDANI AS-5:* .



Obr. 2.13: Přidaní AS s číslem 5

Prvním krokem je vytvoření dalšího uzlu, který bude reprezentovat hraniční směrovač nového AS. Předchozí hodnota vytvořených uzlů se změní z `nodes.Create(4);` na `nodes.Create (5);`.

```
NodeContainer nodes;
nodes.Create (5);
```

Jelikož je směrovač n4 propojen se směrovači `n0` a `n3` je třeba přidat další 2 spoje pro propojení směrovačů - `n4n0` a `n4n3`.

```
//PRIDANI AS-5: Zde doplnte pridani spoje n4n0
NetDeviceContainer n4n0 = pointToPoint.Install(
    nodes.Get (4), nodes.Get (0));
//PRIDANI AS-5: Zde doplnte pridani spoje n4n3
NetDeviceContainer n4n3 = pointToPoint.Install(
    nodes.Get (4), nodes.Get (3));
```

Směrovač - č.AS	Rozhraní	IP adresa	PCAP
n0	1	10.1.1.1	p2p_nodes-0-0.pcap
AS - 1	2	10.0.1.1	p2p_nodes-0-1.pcap
	3	10.4.0.2	p2p_nodes-0-2.pcap
n1	1	10.0.1.2	p2p_nodes-1-0.pcap
AS - 2	2	10.0.2.1	p2p_nodes-1-1.pcap
n2	1	10.0.2.2	p2p_nodes-2-0.pcap
AS - 3	2	10.0.3.1	p2p_nodes-2-1.pcap
n3	1	10.0.3.2	p2p_nodes-3-0.pcap
AS - 4	2	10.1.3.1	p2p_nodes-3-1.pcap
	3	10.4.3.2	p2p_nodes-3-2.pcap
n4	1	10.4.0.1	p2p_nodes-4-0.pcap
AS - 5	2	10.4.3.1	p2p_nodes-4-1.pcap
serverLanNode - AS-1	1	10.1.1.2	p2p_nodes-5-0.pcap
clientLanNode - AS-4	1	10.1.3.2	p2p_nodes-6-0.pcap

Tab. 2.2: Tabulka s výpisem adres a PCAP souborů pro rozhraní směrovačů po přidaní AS-5

Vytvořeným propojením je nutno přidělit jednotlivé rozhraní a adresy z definovaného adresního rozsahu, tak jak je popsáno na obrázku 2.13 nebo dle tabulky 2.2.

```
//PRIDANI AS-5: Zde doplnte vyber adresniho rozsahu
    a prirazeni IP adres pro spoj n4n0
ipv4AddrHelper.SetBase ("10.4.0.0", "255.255.255.252");
Ipv4InterfaceContainer i4i0 = ipv4AddrHelper.Assign (n4n0);
//PRIDANI AS-5: Zde doplnte vyber adresniho rozsahu
    a prirazeni IP adres pro spoj n4n3
ipv4AddrHelper.SetBase ("10.4.3.0", "255.255.255.252");
Ipv4InterfaceContainer i4i3 = ipv4AddrHelper.Assign (n4n3);
```

Po vytvoření uzlu, propojení a přidělení správných adres pro rozhraní, můžeme nastavit příslušné sousedství mezi směrovačmi `n4-n0` a `n4-n3`.

```
//PRIDANI AS-5: Zde doplnte vytvoreni sousedstva mezi n4-n0
quagga.BgpAddNeighbor (nodes.Get(4), "10.4.0.2",
quagga.GetAsn (nodes.Get (0)));
quagga.BgpAddNeighbor (nodes.Get(0), "10.4.0.1",
quagga.GetAsn (nodes.Get (4)));

//PRIDANI AS-5: Zde doplnte vytvoreni sousedstva mezi n4-n3
quagga.BgpAddNeighbor (nodes.Get(4), "10.4.3.2",
quagga.GetAsn (nodes.Get (3)));
quagga.BgpAddNeighbor (nodes.Get(3), "10.4.3.1",
quagga.GetAsn(nodes.Get (4)));
```

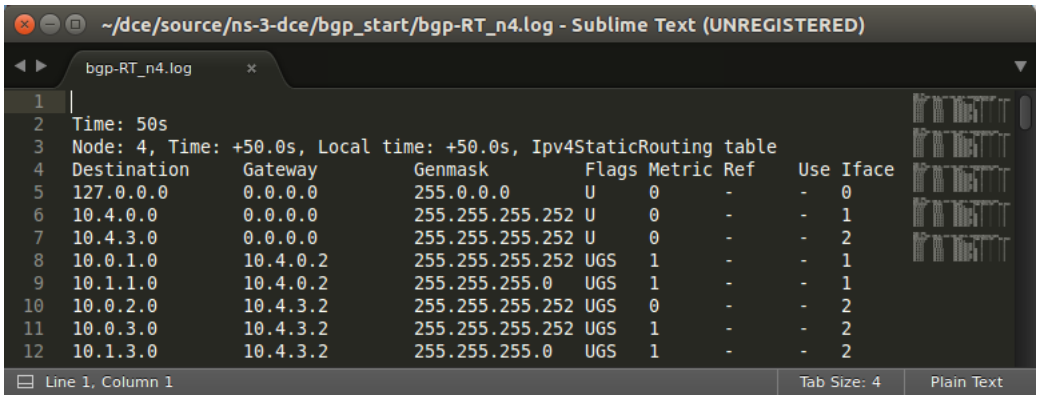
Posledním krokem je získání směrovací tabulky směrovače `n4` a doplněni smerovače do NetAnim animace.

```
//PRIDANI AS-5: Zde doplnte vytvoreni smerovaci tabulky
    pro smerovac n4
Ptr<OutputStreamWrapper> routingTable_n4 =
    Create<OutputStreamWrapper>(
        "bgp_start/bgp-RT_n4.log", std::ios::out);
ipv4RoutingHelper.PrintRoutingTableEvery(
    Seconds(50), nodes.Get(4), routingTable_n4);
```

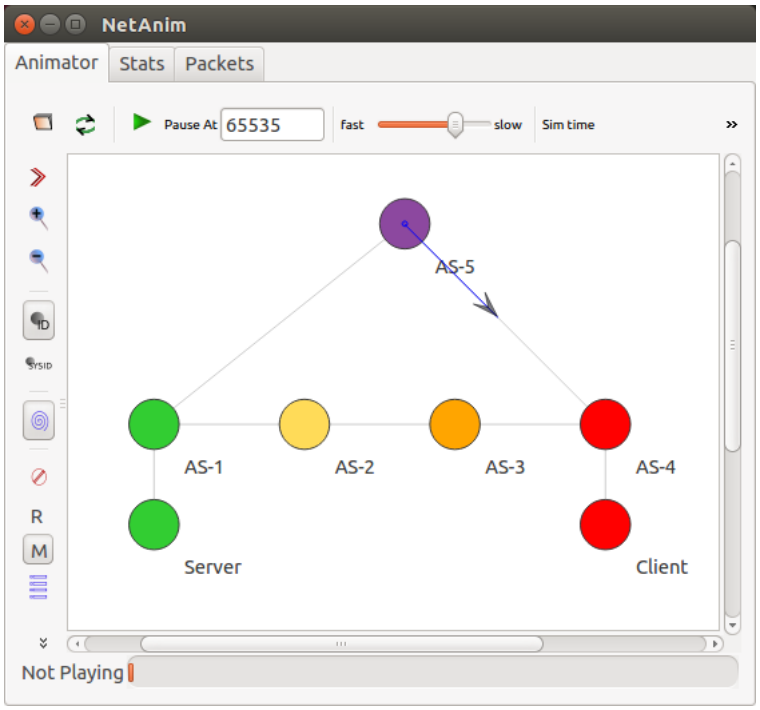
```
//PRIDANI AS-5: Zde doplnte vytvoreni ikony pro smerovac n4
  anim.SetConstantPosition(nodes.Get(4), 6.0, 2.0, 0);
  anim.UpdateNodeDescription(nodes.Get(4), "AS-5");
  anim.UpdateNodeColor(nodes.Get(4), 140, 70, 159);
```

Po spuštění, si v příslušných trasovacích souborech (`p2p_nodes-4-0.pcap` a `p2p_nodes-4-1.pcap`) ověřte, že sousedství byly správně vytvořeny a v souboru `bgp-RT_n4.log` se podívejte, zda daný směrovač získal informace o všech sítích. Směrovací tabulka směrovače `n4` by se měla shodovat s obrázkem 2.14.

Spusťte si animaci (měla by se shodovat s obrázkem 2.15) a ověřte jakým směrem šel provoz UDP paketů. Tento směr by se měl shodovat se směrem vyznačeným šipkami na obrázku 2.16. Po úspěšném vytvoření nových sousedství by si měl protokol BGP vybrat pro posílání provozu cestu přes AS-5, která je o jeden skok kratší, než původní cesta.



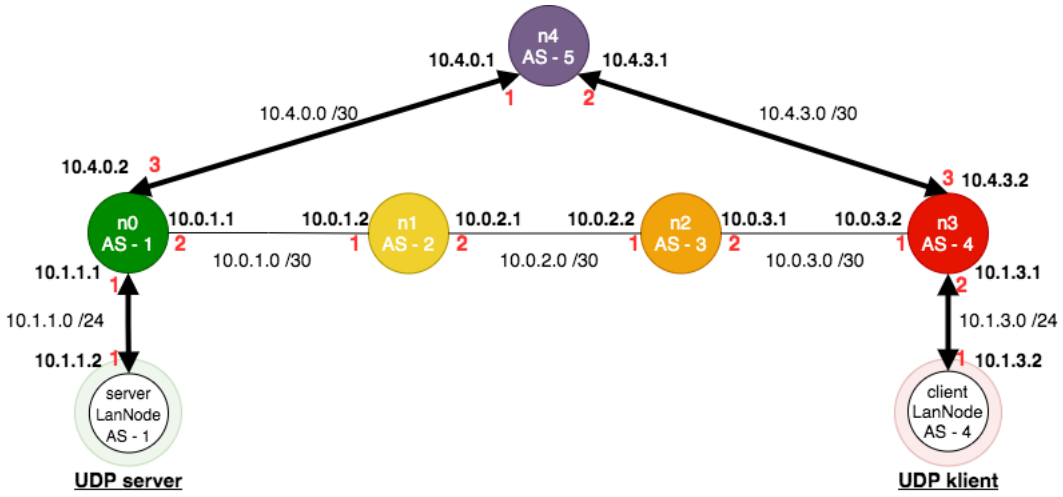
Obr. 2.14: Směrovací tabulka směrovače n4



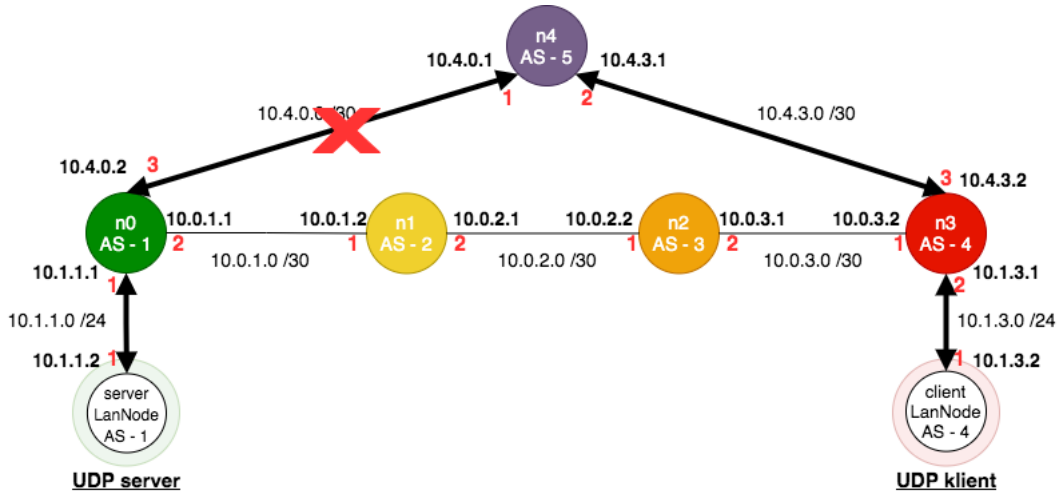
Obr. 2.15: NetAnim animace po přidání směrovače n4

2.1.4 Simulace výpadku

Čtvrtým typem zprávy protokolu BGP je zpráva NOTIFICATION. Touto zprávou dává směrovač vědět, že vypršel časovač Hold Time, kdy neobdržel od svého souseda zprávu KEEPALIVE ani UPDATE nebo nastala jiná chyba ve spojení a dané propojení již nepovažuje za aktivní. Po obdržení této zprávy, zašle směrovač informaci o výpadku svým sousedům již ve formě zprávy UPDATE - Withdrawn Routes, kde specifikuje sítě z jeho rozhraní, které již nepovažuje za aktivní. Tato zpráva se vygeneruje například při simulaci výpadku spojení mezi směrovači `n0` a `n4`, čím bude protokol BGP nucen změnit trasu směrování provozu.



Obr. 2.16: Směr provozu UDP paketu po přidání AS - 5



Obr. 2.17: Znázornění výpadku spoje v simulaci

Simulátor NS3 nemá při definování spoje metodu, která by tento výpadek simulovala, ale je možné nastavit zpoždění daného spoje, které bude dostatečně velké na to, aby ho protokol BGP považoval za výpadek.
K tomuto účelu je třeba přidat funkci `ChangeDelay` na začátek kódu, před funkci `main` na místo označené komentářem `//VYPADEK: Metoda pro simulaci vypadku`. Po zkopírování metody prosím zkontrolujte, aby string `/ChannelList/5/$ns3::PointToPointChannel/Delay` neměl žádné mezery.

```
  void ChangeDelay(){ Config::Set (
      "/ChannelList/5/$ns3::PointToPointChannel/Delay",
            StringValue ("200s"));
}
```

K zjištění, jak se změní směrování sítí je třeba zaslat další množství UDP paketů po době, kdy se stihnou aktualizovat směrovací tabulky směrovačů protokolem BGP. Následující kód je třeba přidat na místo vyznačené komentářem *//VYPADEK: Zde doplnte dalsi posilani paketu:*

```
//VYPADEK: Zde doplnte dalsi posilani paketu:
ApplicationContainer clientApps2 =
    echoClient.Install(clientLanNodes.Get (1));
 clientApps2.Start(Seconds (205.0));
 clientApps2.Stop(Seconds (endTime));
```

Volání funkce `ChangeDelay` je nutné naplánovat na daný čas v simulaci. Následující řádek vložte před krok definování ukončení simulace *Simulator::Stop (Seconds (endTime));* dle označení komentářem *//VYPADEK: Zde doplnte radek pro pridani vypadku:*

```
Simulator::Schedule (Seconds(40), &ChangeDelay);
```

Spusťte znovu simulaci, otevřete si trasovací soubor `p2p_nodes-4-0.pcap`, kde si vyfiltrujte BGP provoz. Funkce `ChangeDelay` nastaví vysoké zpoždění spoje mezi směrovači `n4` a `n0` to je znázorněno na obrázku 2.17.

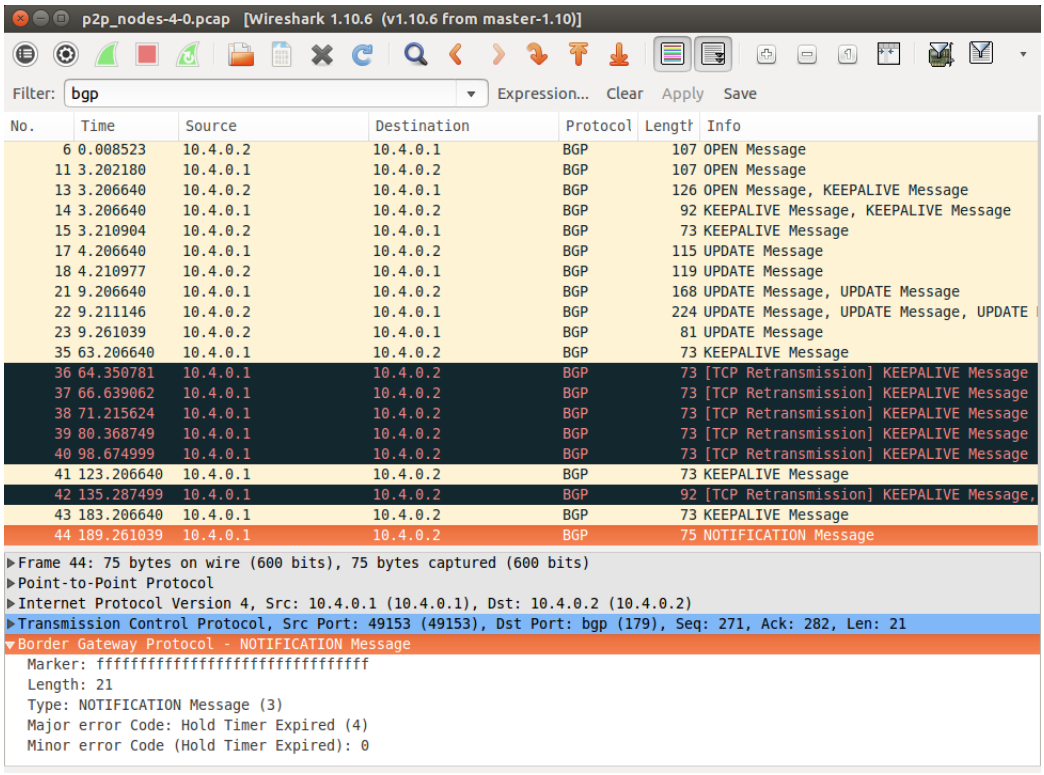
Toto zpoždění způsobí zpoždění zprávy KEEPALIVE od směrovače `n0 - 10.4.0.2`, čímž vyprší HoldTime směrovače n4 sousedství s n0. Směrovač zašle směrovači n0 zprávu NOTIFICATION, která je v souboru `p2p_nodes-4-0.pcap` (obrázek 2.18) o tom, že dané spojení již nepovažuje za aktivní. Tato zpráva v sobě obsahuje i důvod chyby Major/Minor error Code, která nastala. V tomto případě to bylo vypršení časovače HoldTime.

Směrovač `n0` posíla zprávu UPDATE - Withrawn Routes dalším sousedům viz. obrázek 2.19 nebo soubor `p2p_nodes_0_1` o tom, že se daná trasa (přes síť 10.4.3.0/30 do 10.1.3.0/24) již není aktivní.

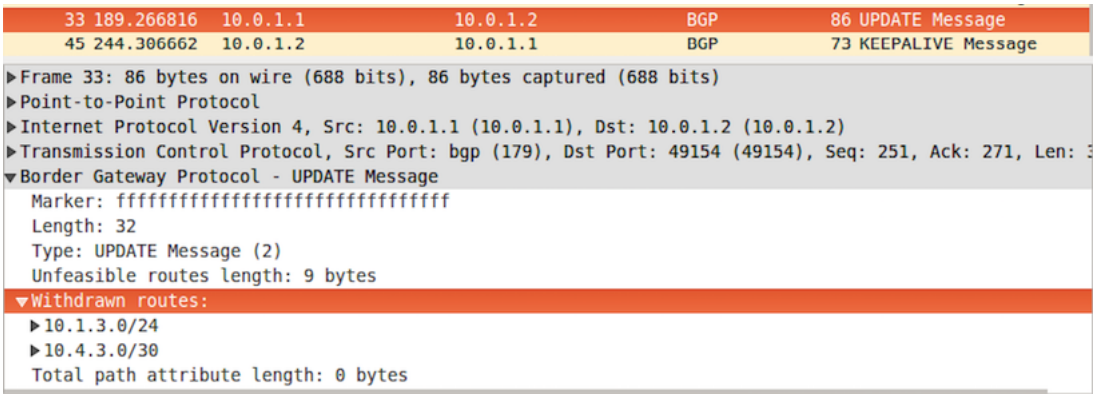
Zprávou UPDATE - Withrawn Routes se změní směrovací tabulky směrovačů. Soubor `bgp-RT_n3.log` obsahuje směrovací tabulku směrovače n3, daná změna jeho směrovací tabulky je na obrázku 2.20.

Směrovač n3 před výpadkem spojení směrovačů n1 - n5 provoz pro síť 10.1.1.0/24 posílal přes rozhraní číslo 3, po tom jak nastal výpadek a přišla mu další zpráva Withdrawn Routes, změnil tento záznam, dle dat, které mu dosud byly doručeny a považuje je za aktivní. Provoz pro tuto síť bude již posílat přes rozhraní číslo 1, jako na obrázku 2.22.

Spusťte znovu NetAnim simulaci, kde je v čase 72 vidět zastavení paketů 2.21, přeposílání UPDATE zpráv čím se protokol BGP adaptuje pro změnu trasy. Následné posílání UDP provozu v čase 205 pak probíhá už přes změněnou trasu, kterou protokol BGP našel.



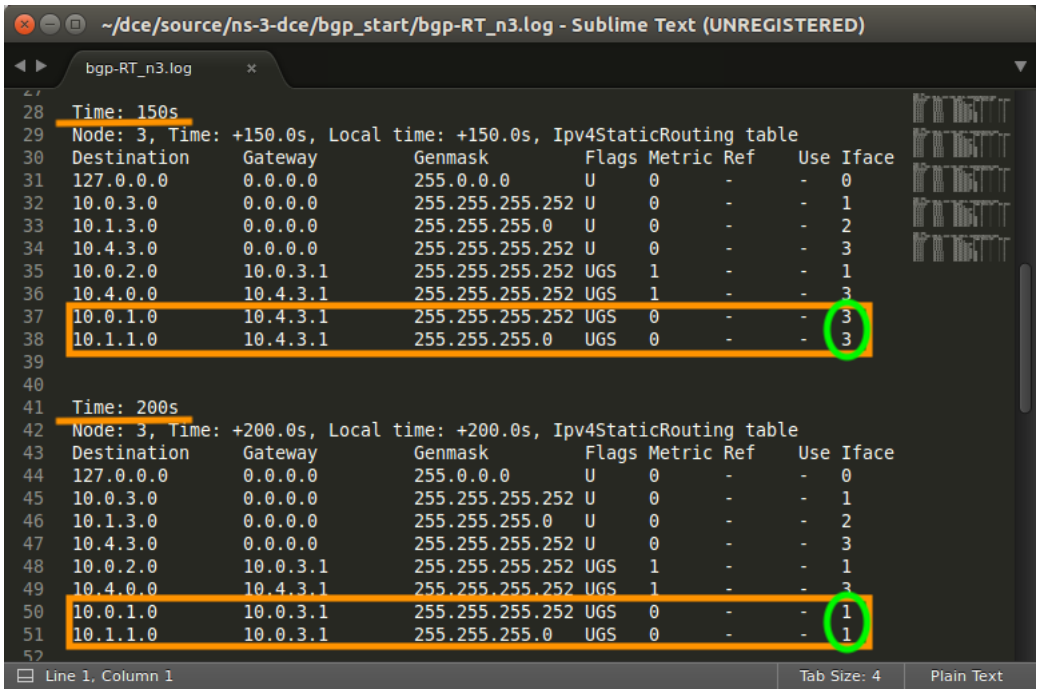
Obr. 2.18: Zpráva NOTIFICATION na směrovači n4 rozhraní číslo 1



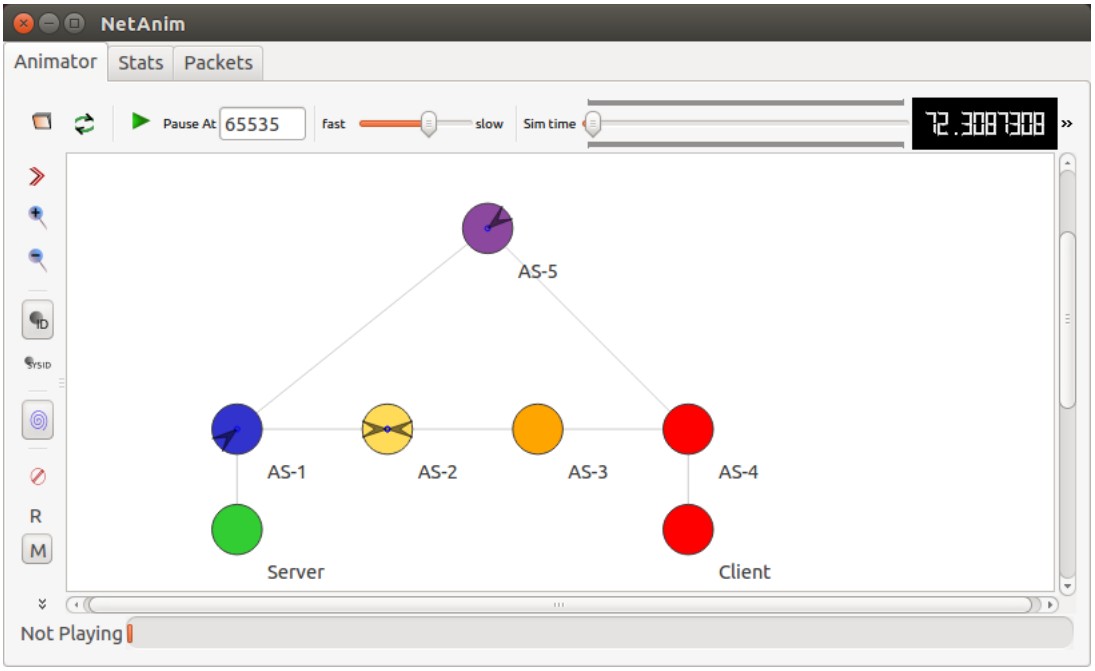
Obr. 2.19: Zpráva UPDATE Withdrawn Routes na směrovači n0 rozhraní číslo 2

2.1.5 Rozložení provozu - PeerLink

Jelikož protokol BGP slouží k směrování provozu mezi AS, množství tohoto provozu je poměrně veliké. K lepšímu rozložení provozu mu slouží různé nastavitelné parametry jako např. MED nebo LOCAL_PREF pomocí kterých lze dále upravovat směrovací politika. Bohužel ani nástavba Quagga definování těchto parametrů pro simulátor NS3 nepodporuje. Rozložení je však možné přizpůsobit přidáním tzv. PeerLinku. Nastavení PeerLinku mezi směrovači způsobí to, že jeden ze sousedů nebude

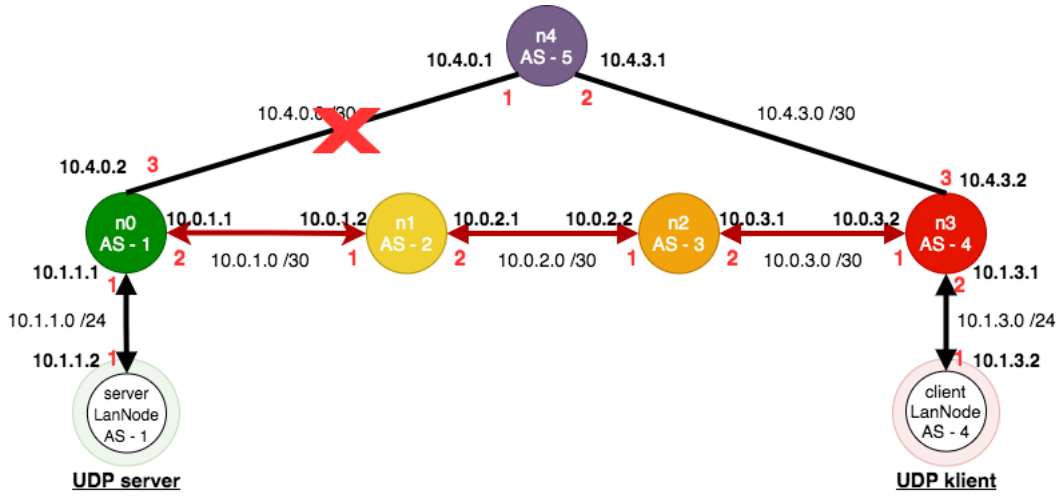


Obr. 2.20: Změna směrovací tabulky směrovače n3 po simulaci výpadku spojeni



Obr. 2.21: Simulace výpadku v animaci NetAnim

zasílat druhému zprávy typu UPDATE, čím nebude vědět o možných jiných trasách. Nejprve v kódu zakomentujte nastavení výpadku a pozdější přeposílání UDP paketů.

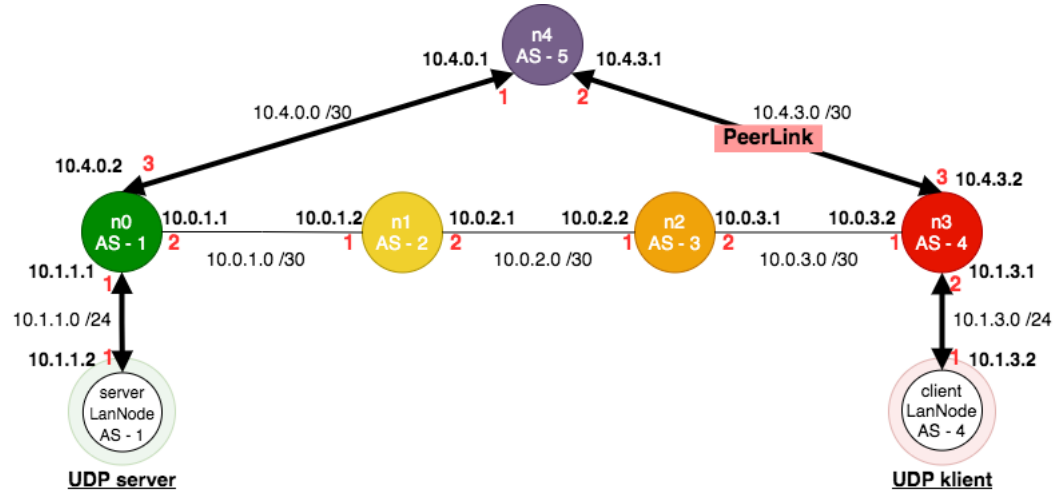


Obr. 2.22: Změna směru provozu po simulaci výpadku spojení

```
//PEERLINK: Zakomentovat vypadek
//Simulator::Schedule (Seconds(40), &ChangeDelay);
```

```
//PEERLINK: Zakomentovat pozdější přeposílání UDP paketů
//ApplicationContainer clientApps2 =
        echoClient.Install(clientLanNodes.Get (1));
//clientApps2.Start(Seconds (205.0));
//clientApps2.Stop(Seconds (endTime));
```

Provoz tedy probíhá přes AS-5, jak je znázorněno šípkami na obrázku 2.23.



Obr. 2.23: Směr provozu UDP paketů před přidáním PeerLinku

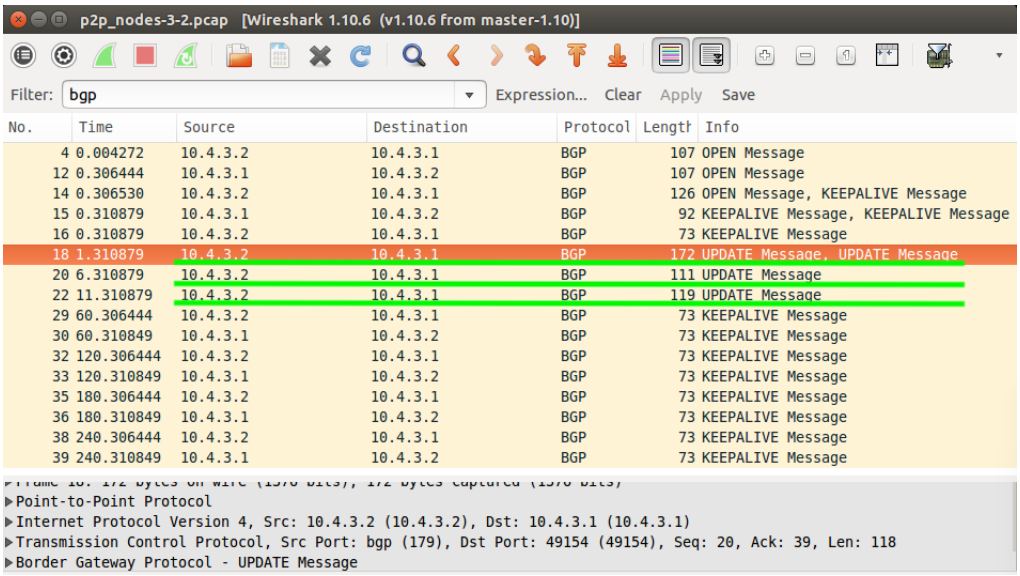
Směr tohoto provozu, ale můžeme změnit pomocí metody QuaggaHelper::AddPeerLink() tak, aby na nějakém ze spojů neprobíhal provoz oboustranně, ale jen jedním směrem. Například provoz od klienta k serveru bude procházet přes směrovač *n2* a *n1*, a provoz od serveru ke klientovi, přes směrovač *n4*. To je docíleno přidáním následujícího řádku po definování sousedství mezi směrovači n4 a n3. Místo přidání řádku je vyznačeno komentářem *//PEERLINK: Zde doplne pridani PeerLinku na smerovaci n4.*

```
//PEERLINK: Zde doplne pridani PeerLinku na smerovaci n4
quagga.BgpAddPeerLink (nodes.Get (4), "10.4.3.2");
```

Pro výraznější vizualizaci této změny posuňte čas začátku posílání paketů na 23s kdy budou mít oba směrovače všechny informace pro správné přesměrování provozu.

```
//PEERLINK: Zmena casu na 23s
clientApps.Start(Seconds (23.0));
```

Tímto nebude směrovač `n4` zasílat UPDATE zprávy směrovači `n3`, ale směrovač `n3` bude o UPDATE zprávách směrovač `n4` informovat. To, že směrovač `n3` jenom posílá zprávy UPDATE a žádné zprávy UPDATE od `n4` nepřijímá, je možné potvrdit v trasovacím souboru `p2p_nodes-3-2.pcap` 2.24.



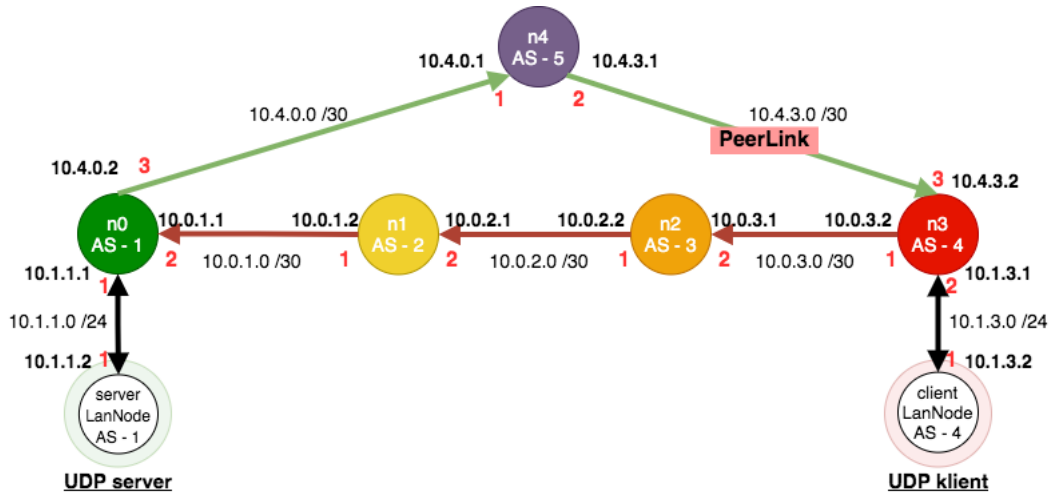
Obr. 2.24: Přidání PeerLinku mezi směrovače n4 a n3

Jelikož směrovač `n4` neposlal směrovači `n3` zprávu o tom, že má síť 10.1.1.0/24 k dispozici, pošle směrovač `n3` provoz ve směru od klienta na směrovač `n2`. Tímto se vlasností BGP sousedství snížila zátěž směrovače `n4`, protože už přes něj probíhá jen provoz od serveru.

Spusťte NetAnim simulaci pro ověření směru provozu:

- provoz od klienta ke serveru by měl probíhat přes směrovače `n3`, `n2`, `n1`, a `n0`
- provoz od serveru ke klientovi by měl probíhat přes směrovače `n0`, `n4`, a `n3`

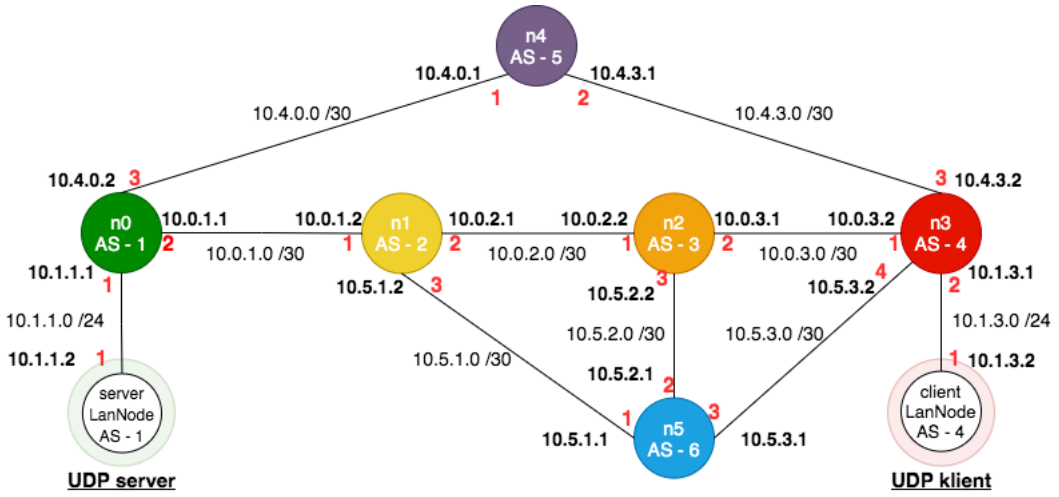
Směr provozu po přidání PeerLinku mezi směrovače n4 a n3 je znázorněn na obrázku. 2.25.



Obr. 2.25: Změna trasy provozu po přidání PeerLinku mezi směrovači n4 a n3

2.1.6 Přidání dalšího AS - samostatný úkol

Samostatnou částí je přidání směrovače `n5`, který bude reprezentovat AS - 6 tak, aby výsledná topologie odpovídala obrázku 2.26.



Obr. 2.26: Vysledná topologie po přidání AS-6

Místa v kódu, kam je potřeba vložit jednotlivé doplnění, jsou označeny komentářem *//PRIDANI AS-6*. Jako první je třeba zvýšit počet uzlů na 6.

```
//PRIDANI AS-6: Zvysit pocet uzlu pro n5 na 6
nodes.Create (6);
```

Dále je třeba přidat spoje pro připojení uzlu `n5` k uzlům `n1`, `n2` a `n3`.

```
//PRIDANI AS-6: Vyber adresniho rozsahu a priradzeni IP adres
    pro spoj n5n1
NetDeviceContainer n5n1 =
    pointToPoint.Install(nodes.Get (5), nodes.Get (1));
//PRIDANI AS-6: Vyber adresniho rozsahu a priradzeni IP adres
    pro spoj n5n2
NetDeviceContainer n5n2 =
    pointToPoint.Install(nodes.Get (5), nodes.Get (2));
//PRIDANI AS-6: Vyber adresniho rozsahu a priradzeni IP adres
    pro spoj n5n3
NetDeviceContainer n5n3 =
    pointToPoint.Install(nodes.Get (5), nodes.Get (3));
```

Pro vytvořené spoje je třeba vybrat adresní prostor a přiřadit uzlům dané adresy dle obrázku 2.26.

```
//PRIDANI AS-6: Vyber adresniho rozsahu a priradzeni IP adres
    pro spoj n5n1
ipv4AddrHelper.SetBase ("10.5.1.0", "255.255.255.252");
Ipv4InterfaceContainer i5i1 = ipv4AddrHelper.Assign (n5n1);
//PRIDANI AS-6: Vyber adresniho rozsahu a priradzeni IP adres
    pro spoj n5n2
ipv4AddrHelper.SetBase ("10.5.2.0", "255.255.255.252");
Ipv4InterfaceContainer i5i2 = ipv4AddrHelper.Assign (n5n2);
//PRIDANI AS-6: Vyber adresniho rozsahu a priradzeni IP adres
    pro spoj n5n3
ipv4AddrHelper.SetBase ("10.5.3.0", "255.255.255.252");
Ipv4InterfaceContainer i5i3 = ipv4AddrHelper.Assign (n5n3);
```

Vaším úkolem je přidání vytvoření jednotlivých BGP sousedství mezi směrovači podle adres v tabulce 2.3 a obrázku 2.26.

Místa kam je potřeba vytvoření sousedství jsou označeny komentářem *//PRIDANI AS-6: Pridani sousedstvi*. Nastavení PeerLinku mezi směrovači `n3` a `n4` nechte aktivní.

```
//PRIDANI AS-6: Pridani sousedstvi mezi n5-n1
//PRIDANI AS-6: Pridani sousedstvi mezi n5-n2
//PRIDANI AS-6: Pridani sousedstvi mezi n5-n3
```

Dále doplňte získání směrovací tabulky a ikonu pro simulaci v NetAnim pro uzel n5.

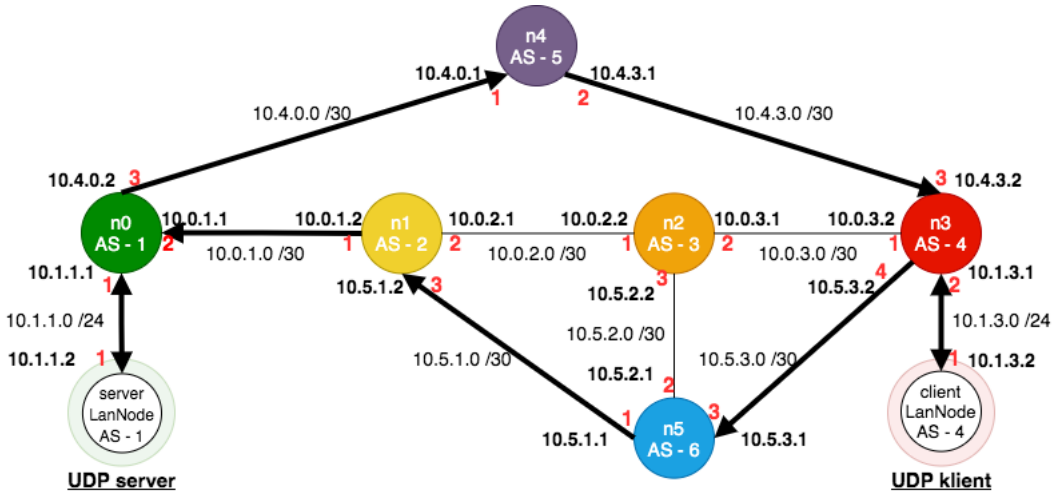
```
//PRIDANI AS-6: Pridani smerovaci tabulky pro n5
Ptr<OutputStreamWrapper> routingTable_n5 =
    Create<OutputStreamWrapper>(
        "bgp_start/bgp-RT_n5.log", std::ios::out);
ipv4RoutingHelper.PrintRoutingTableEvery(
    Seconds(50), nodes.Get(5), routingTable_n5);
```

```
//PRIDANI AS-6: Přidání ikony pro směrovač n5 v NetAnim
  anim.SetConstantPosition(nodes.Get(5), 7.0, 10.0, 0);
  anim.UpdateNodeDescription(nodes.Get(5), "AS-6");
  anim.UpdateNodeColor(nodes.Get(5), 0, 0, 255);
```

Po spuštění simulace si zobrazte vytvořené trasovací soubory, směrovací tabulky a animaci. Pomocí nich ověřte zda se směr provozu shoduje se směrem šipek na obrázku 2.27.

Směrovač - č.AS	Rozhraní	IP adresa	PCAP
n0	1	10.1.1.1	p2p_nodes-0-0.pcap
AS - 1	2	10.0.1.1	p2p_nodes-0-1.pcap
	3	10.4.0.2	p2p_nodes-0-2.pcap
n1	1	10.0.1.2	p2p_nodes-1-0.pcap
AS - 2	2	10.0.2.1	p2p_nodes-1-1.pcap
	3	10.5.1.2	p2p_nodes-1-2.pcap
n2	1	10.0.2.2	p2p_nodes-2-0.pcap
AS - 3	2	10.0.3.1	p2p_nodes-2-1.pcap
	3	10.5.2.2	p2p_nodes-2-2.pcap
n3	1	10.0.3.2	p2p_nodes-3-0.pcap
AS - 4	2	10.1.3.1	p2p_nodes-3-1.pcap
	3	10.4.3.2	p2p_nodes-3-2.pcap
	4	10.5.3.2	p2p_nodes-3-3.pcap
n4	1	10.4.0.1	p2p_nodes-4-0.pcap
AS - 5	2	10.4.3.1	p2p_nodes-4-1.pcap
n5	1	10.5.1.1	p2p_nodes-5-0.pcap
AS - 6	2	10.5.2.1	p2p_nodes-5-1.pcap
	3	10.5.3.1	p2p_nodes-5-1.pcap
serverLanNode - AS-1	1	10.1.1.2	p2p_nodes-6-0.pcap
clientLanNode - AS-4	1	10.1.3.2	p2p_nodes-7-0.pcap

Tab. 2.3: Tabulka s výpisem adres a PCAP souborů pro rozhraní směrovačů po přidání AS-6



Obr. 2.27: Vysledná topologie po přidání AS-6

Provoz probíhá podobně jako tomu bylo při nastaveném PeerLink spojení, ale směrem k serveru BGP protokol upřednostnil nově vytvořené spojení přes směrovač `n5`, než původní spojení přes směrovač `n2`.

2.1.7 Kontrolní otázky

1. Co je to AS a co je jeho identifikátorem?
2. Jaký je rozdíl mezi IGP protokoly a EGP protokolem?
3. Kolik typů zpráv má protokol BGP?
4. Jakými zprávami se definuje sousedství mezi BGP směrovači?
5. Co znamená NLRI a v které zprávě BGP se dá najít?

2.2 Úloha: Zprávy protokolu ICMPv6

2.2.1 Úvod

Tato úloha se zabývá strukturou a funkcionalitou zpráv protokolu ICMPv6 (Internet Control Message Protokol pro IP verze 6). Protokol ICMPv6 používají IPv6 uzly k hlášení chyb při přenosu paketů či pro provádění dalších funkcí internetové vrstvy, například pro diagnostiku sítě či vyhledávání směrovačů. ICMPv6 je nedílnou součástí sady IPv6 a musí být implementován pro každý IPv6 uzel.
Protokol ICMPv6 navíc slouží jako framework pro protokoly MLD (Multicast Listener Discovery) a ND (Neighbor Discovery), které provádějí úkoly přenosu informací o členství v multicastových skupinách (ekvivalent protokolu IGMP v IPv4) a překlad IP adres (ekvivalent protokolu ARP v IPv4).
Hlavičku ICMPv6 identifikuje číslo 58 v poli NextHeader value předchozí hlavičky. ICMPv6 zprávy obsahují pole:

- *Typ (Type)* - označuje typ zprávy. Jeho hodnota rozhoduje o formátu zbývajících dat, které zpráva obsahuje.
- *Kód (Code)* - používá se pro doplnění detailů zprávy.
- *Kontrolní součet (Checksum)* - pole pro k detekci poškození dat v ICMPv6 zprávě a v částech IPv6 hlavičky.
- *Tělo zprávy (Message Body)* - obsahuje samotný obsah zprávy, jejíž struktura závisí na typu zprávy.

Na základě funkcionalit protokolu ICMPv6 jsou jeho zprávy rozděleny do dvou skupin na:

- *Chybové zprávy* - jsou označeny v poli Type hodnotami 0 - 127. Patří zde například zprávy - Destination unreachable (Typ 1), Packet Too Big (Typ 2) či Time Exceeded (Typ 3).
- *Informační zprávy* - jsou označeny v poli Type hodnotami 128 - 255. Jsou to například zprávy Echo Request (Typ 128)/ Echo Reply (Typ 129) nebo zpráva Neighbor Solicitation (Typ 135) a další.

Popis úlohy

Tato úloha je rozdělena na 3 části. V každé části budou ukázány a popsány různé zprávy protokolu ICMPv6, případně jejich variace.

1. V první části jsou popsány zprávy související s konfigurací IPv6 adres (informační zprávy): NS - Neighbor Solicitation, NA - Neighbor Advertisement, RS - Router Solicitation a RA - Router Advertisement.
2. V druhé části je uvedena informační zpráva spojena se směrováním v IPv6 - zpráva Redirect a funkce dvojice zpráv NA/NS jako náhrada ARP pro IPv6 a zprávy aplikace Ping - Echo Request/Echo Reply.
3. Ve třetí části jsou uvedeny nejčastější chybové zprávy ICMPv6 - zpráva Destiantion unreachable, zpráva Time Exceeded a zpráva Packet Too Big.

2.2.2 1. část: Konfigurace IPv6 adres

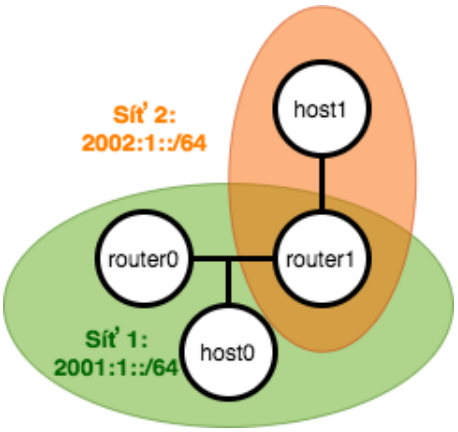
Protokol IPv6 nabízí několik možnosti konfigurace globálních IPv6 adres. Podobně jako u protokolu IPv4, i u IPv6 je možné adresy přidělit staticky nebo stavovou konfigurací. Protokol IPv6 k těmto dvěma možnostem ještě přidává třetí možnost - *Bezstavovou konfiguraci adres - SLAAC (Stateless Address Auto-Configuration).* Pro řízení konfiguraci globálních adres je využíván právě protokol ICMPv6 a jeho zprávy.
Každý ze způsobů konfigurace adres používá jistou variaci informačních zpráv protokolu ICMPv6, přesněji jeho rozšíření v podobě NDP (Neighbor Discovery Protocol), pro který byly tyto zprávy navrženy. V rámci první části úlohy bude popsána statická a bezstavová konfigurace globálních adres.

- *Statická konfigurace* - používá zprávu Neigbhor Solicitation, kterou vyvolá samotný uzel pro potvrzení mechanismu DAD - Duplicate Address Detection
- *Bezstavová konfigurace SLAAC (Stateless Address Auto-Configuration)* - používá dvojici zpráv RS/RA - Router Solicitation Router Advertisement, které nesou informace pro konfiguraci globální adresy.
- *Stavová konfigurace* - není v této úloze popsaná. Také využívá zprávy RS/RA, kde jsou pro uzly uvedeny informace o možnosti konfigurace za pomocí DHCPv6.

Popis vytvořeného souboru

V úloze je použita topologie uvedena na obrázku 2.28. Topologie se skládá ze dvou sítí - Síť 1 s prefixem `2001:1::/64` a Síť 2 s prefixem `2002:1::/64`. V Síti 1 se nacházejí 3 uzly - uzel `host0`, který představuje koncové zařízení sítě a uzly `router0` a `router1`, které reprezentují směrovače. V síti 2 se nacházejí 2 uzly - `host1` představující koncové zařízení a směrovač `router1`, který je součástí obou sítí. Celá tato topologie je definována v souboru `icmpv6.cc`, který budete postupně podle uvedených pokynů upravovat.

Na začátku souboru jsou definovány moduly potřebné pro simulaci, `namespace` kde má program najít použité funkce a název komponenty, pomocí které má program



Obr. 2.28: Základní topologie úlohy

vypisovat pomocné zprávy.

```
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/csma-module.h"
        ...
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("Icmpv6");
```

Dále je v kódu uvedena implementace metody `printIpAddresses` pro výpis adres jednotlivých rozhraní uzlu do konzoly programu. Tato metoda bude volána později v kódu pro zobrazení adres které byly přiděleny uzlům pro jejich rozhraní. Její přesná implementace není pro tento úkol podstatná a nebude se v rámci úkolu nijak měnit.

```
void printIpAddresses (Ptr<Node>& n){
... }
```

Na začátku funkce `main()` jsou vytvořeny proměnné typu `Node`, které reprezentují každý z uzlů topologie 2.28. Ty jsou dále sdruženy do proměnné tipu `NodeContainer` podle sítí, do jakých patří.

```
  Ptr<Node> host0 = CreateObject<Node> ();
  Ptr<Node> router0 = CreateObject<Node> ();
  Ptr<Node> router1 = CreateObject<Node> ();
  Ptr<Node> host1 = CreateObject<Node> ();
  NodeContainer net1 (host0, router0, router1);
  NodeContainer net2 (router1, host1);
  NodeContainer all (host0, router0, router1, host1);
```

Pomocí třídy `CsmaHelper` je definován typ propojení uzlů. Propojení má definováno dva parametry - *přenosovou rychlost* a *zpoždění*. Toto propojení je následně nainstalováno na jednotlivé uzly podle sítí, do kterých patří. Tímto budou mít uzly připraveny síťové karty s MAC adresami.
Síťové karty uzlů jsou uloženy v proměnné typu `NetDeviceContainer` a MAC adresy jsou jim přiděleny v pořadí, v jakém jsou na jednotlivé uzly nainstalované propojení. Například jako první je propojení definované pro síť 1. Ta je reprezentována proměnnou `net1`. Jako první byl do této proměnné vložený uzel `host0`, ten bude tedy mít MAC adresu `00:00:00:00:00:01`.

```
CsmaHelper csma;
csma.SetChannelAttribute("DataRate", DataRateValue(5000000));
csma.SetChannelAttribute("Delay", TimeValue(MilliSeconds(2)));
NetDeviceContainer netDevices_net1 = csma.Install (net1);
NetDeviceContainer netDevices_net2 = csma.Install (net2);
```

Aby mohly uzly komunikovat pomocí sady protokolů IPv6 potřebují mít nainstalován IPv6 stack. O to se postará třída `InternetStackHelper`. Metoda `SetIpv4StackInstall()` s parametrem `false` definuje, že nebude využit IPv4 stack a tedy uzly budou využívat stack pro IPv6.

```
InternetStackHelper internetv6;
internetv6.SetIpv4StackInstall (false);
internetv6.Install (all);
```

Komentáře *//Samostatná část* pro samostatnou část zatím přeskočte.

Definování konfigurace adres

Před samotným definováním konfigurace adres jsou vytvořeny proměnné `net1_address` a `net2_address`, které uchovávají hodnotu prefixů jednotlivých sítí.

```
Ipv6AddressHelper ipv6;
Ipv6Address net1_address = Ipv6Address ("2001:1::");
Ipv6Address net2_address = Ipv6Address ("2002:1::");
```

Jak bylo uvedeno na začátku úlohy, uzly mohou mít nakonfigurovány globální adresy buď staticky nebo pomocí bezstavové konfigurace - SLAAC. Staticky nakonfigurované adresy bude mít směrovač `router1` a uzly `host0`, `router0` a `host1` použijí bezstavovou konfiguraci SLAAC.

Statická konfigurace adres

Staticky nakonfigurované adresy bude mít směrovač `router1`. Jako první, je v kódu nakonfigurováno jeho rozhraní k síti 1. Pro tuto konfiguraci je použita proměnná typu `Ipv6AddressHelper ipv6` pro kterou je metodou `SetBase (net1_address, Ipv6Prefix (64))` definováno, aby byl pro vytvoření adresy použit prefix `2001:1::`. Dále je do proměnné `device_router1_net1` vložena síťová karta uzlu `router1` a pomocí metody `Assign()` se na jejím rozhraní vytvoří link-local a globální IPv6 adresa pomocí metody EUI-64. To, zda je daný uzel směrovačem nebo koncovým uzlem, určuje metoda `SetForwarding (0, true)`, kde hodnota `true značí`, že se nejedná o uzel, ale o směrovač.

```
ipv6.SetBase (net1_address, Ipv6Prefix (64));
NetDeviceContainer device_router1_net1;
device_router1_net1.Add(netDevices_net1.Get(2));
Ipv6InterfaceContainer interface_router1_net1 =
    ipv6.Assign(device_router1_net1);
interface_router1_net1.SetForwarding(0, true);
```

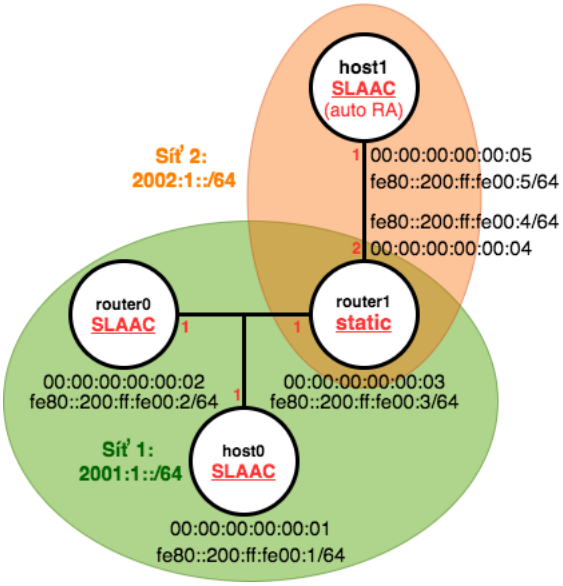
Pro druhé rozhraní směrovače `router1` je konfigurace adres podobná. Je však třeba zvolit nový prefix a druhé síťové rozhraní.

```
ipv6.SetBase (net2_address, Ipv6Prefix (64));
NetDeviceContainer device_router1_net2;
device_router1_net2.Add(netDevices_net2.Get(0));
Ipv6InterfaceContainer interface_router1_net2 =
    ipv6.Assign(device_router1_net2);
interface_router1_net2.SetForwarding(0, true);
```

Tímto budou pro rozhraní směrovače `router1` nakonfigurovány globální a link-local adresy. Link-local adresu by si měl mít možnost přidělit každý uzel i sám hned po startu rozhraní, protože bez ní uzel nemůže komunikovat ani v rámci své sítě. Ekvivalentem link-local adresy v IPv4 je adresa z rozsahu `169.254.0.0/16`. V rámci IPv6 je pro link-local adresy vyhrazen rozsah `fe80::/10`. Link-local i globální adresy simulátor NS3 generuje pomocí metody EUI-64, tedy na základě MAC adresy daného uzlu. Na obrázku 2.29 jsou popsány MAC adresy a příslušné link-local adresy jednotlivých uzlů.

Před konfigurací globálních adres jsou pro zbývající uzly - `host0`, `router0` a `host1` definované pouze jejich rozhraní pomocí metody `AssignWithoutAddress()`, které zatím získají jenom link-local adresy.

```
NetDeviceContainer device_host0;
device_host0.Add(netDevices_net1.Get(0));
```



Obr. 2.29: MAC a link-local adresy uzlů

```
Ipv6InterfaceContainer interface_host0 =
    ipv6.AssignWithoutAddress(device_host0);
NetDeviceContainer device_router0;
device_router0.Add(netDevices_net1.Get(1));
Ipv6InterfaceContainer interface_router0 =
    ipv6.AssignWithoutAddress(device_router0);
NetDeviceContainer device_host1;
device_host1.Add(netDevices_net2.Get(1));
Ipv6InterfaceContainer interface_host1 =
    ipv6.AssignWithoutAddress(device_host1);
```

Bezstavové přidělování pomocí aplikace RADVD

Pro získání globální adresy bezstavovou konfigurací budou uzly potřebovat získat prefix, s jakým si tuto adresu mají vytvořit. Tento prefix jim poskytne směrovač `router1`, na kterém bude spuštěna aplikace RADVD.
Tato aplikace bude do sítí zasílat zprávy *RA - Router Advertisement*, v nichž bude pro uzly `host0`, `router0` a `host1` daný prefix specifikován. Do sítě 1 bude `router1` zprávu RA zasílat až po její vyžádání zprávou *RS - Router Solicitation*. Do sítě 2 bude zpráva zasílaná automaticky bez nutnosti její vyžádání pomocí zprávy RS.
Jako první je pro aplikaci RADVD definováno na jaké rozhraní má zasílat jaké prefixy. Jelikož směrovač `router1` propojuje obě sítě, na rozhraní se sítí 1, bude zasílat zprávy RA s prefixem pro síť 1 - `net1_address`, a na rozhraní se sítí 2, bude zasílat zprávy RA s prefixem pro síť 2 - `net2_address`.

To, aby směrovač `router1` nezasílal zprávy RA do sítě 1 automaticky a má čekat na výzvu RS od uzlů `host0` a `router0`, je definováno metodou v posledním řádku následujícího kódu.

```
RadvdHelper radvdHelper;
radvdHelper.AddAnnouncedPrefix(
    interface_r1_net1.GetInterfaceIndex(0), net1_address, 64);
radvdHelper.AddAnnouncedPrefix(
    interface_r1_net2.GetInterfaceIndex(0), net2_address, 64);
radvdHelper.GetRadvdInterface(
interface_r1_net1.GetInterfaceIndex(0))->SetSendAdvert (false);
```

Pro aplikaci je nutno uvést na jakém směrovači (`router1`) má být spuštěna spolu s časem jejího spuštění a ukončení.

```
ApplicationContainer radvdApps = radvdHelper.Install(router1);
radvdApps.Start (Seconds (1.0));
radvdApps.Stop (Seconds (10.0));
```

Získání výstupů

Výstupem tohoto úkolu budou soubory s příponou `pcap`, které budou obsahovat pakety jednotlivých rozhraní uzlů a konzolový výpis ve kterém budou uvedeny adresy, které byly přiděleny rozhraním v čase 0s a 2s.

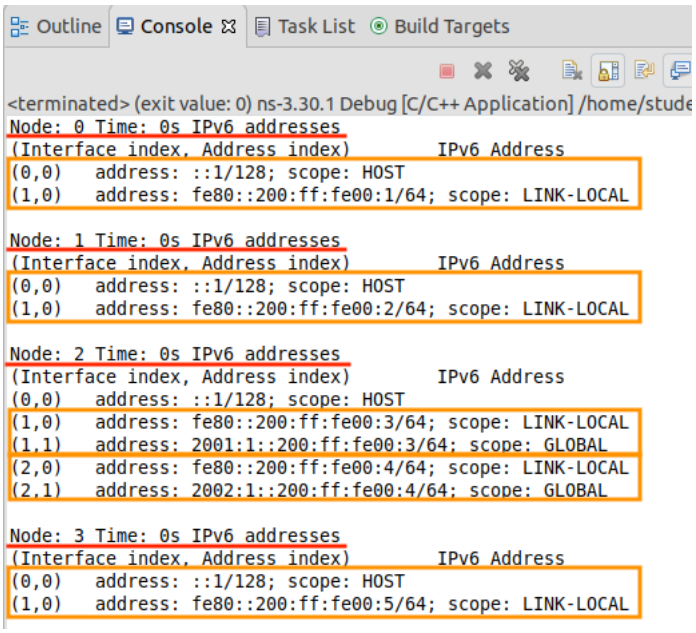
```
csma.EnablePcapAll ("icmpv6");
IpAddressHelper ipAddressHelper;
Simulator::Schedule(Seconds(0.0), &printIpAddresses, host0);
Simulator::Schedule(Seconds(0.0), &printIpAddresses, router0);
    ...
Simulator::Schedule(Seconds(2.0), &printIpAddresses, router1);
Simulator::Schedule(Seconds(2.0), &printIpAddresses, host1);
```

Na závěr je v kódu definováno spuštění samotné simulace.

```
NS_LOG_INFO ("Run␣Simulation.");
Simulator::Stop(Seconds (30));
Simulator::Run ();
Simulator::Destroy ();
NS_LOG_INFO ("Done.");
```

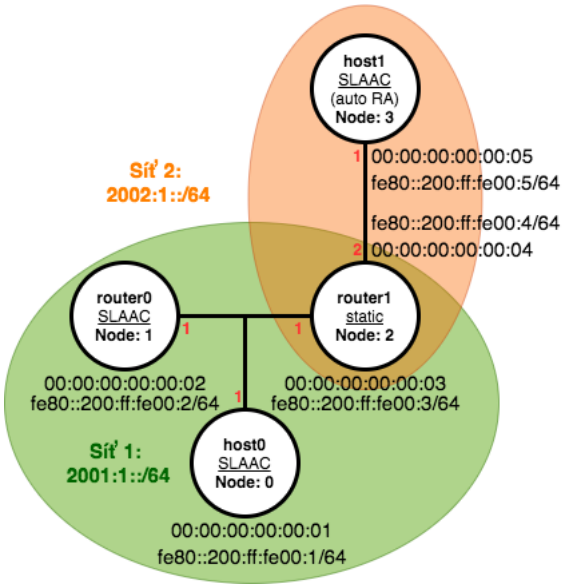
2.2.3 Analýza výstupů pro konfigurace IPv6 adres

Po spuštění a dokončení simulace by se vám měl v konzolovém výstupu zobrazit výsledek přidělování adres podobný obrázku 2.30. Výsledný konzolový výstup bude o něco delší, protože obsahuje i adresy uzlů v čase 2s, které na obrázku 2.30 zachyceny nejsou. Porovnejte si váš výsledný konzolový výstup pro čas 2s s adresami uzlů na



Obr. 2.30: Konzolový výstup po dokončení simulace pro základní topologii

obrázku 2.31. Jednotlivé přidělené adresy by měly shodovat.



Obr. 2.31: Topologie s uvedenými adresami uzlů, které jim byly přiděleny

Ve výstupu se podívejte jaké typy adres se v něm vyskytují. K lepšímu porozumění účelu těchto adres je v tabulce 2.4 shrnutí IPv6 adres s jejich popisem a porovnáním k ekvivalentním adresám v protokolu IPv4. Kdy a jak byly tyto adresy získané během simulace, bude popsáno v následujících částech.

IPv6 adresa	**Účel / Dosah**	**IPv4 ekvivalent**
::/0	Výchozí cesta	0.0.0.0/0
::/128	Nespecifikovaná adresa	0.0.0.0
::1/128	Loopback adresa / Rozhraní uzlu	127.0.0.1/8
fe80::/64	Link-local adresa / LAN	169.254.0.0/16
fc00::/7	Unique-local adresa / LAN	10.0.0.0/8 172.16.0.0/12 192.168.0.0/16

Tab. 2.4: Tabulka s popisem účelu a dosahu nejčastějších IPv6 adres

Statické přidělení adres

Staticky nakonfigurované globální adresy měl směrovač `router1`. Všechny jeho odeslané a přijaté pakety se nacházejí v souborech `icmpv6-2-0.pcap` a `icmpv6-2-1.pcap`, které najdete ve složce `ns-3.30.1` (v ní se nachází i složka scratch), celá cesta k této složce je `/home/student/ns-allinone-3.30.1/ns-3.30.1`. Přípona 2-0 značí, že daný soubor obsahuje pakety pro uzel č. 2 a jeho první rozhraní, tedy rozhraní směrovače `router1` do sítě 1. Otevřete si soubor `icmpv6-2-0.pcap`. Obsah souboru by měl shodovat s obsahem na obrázku 2.32.

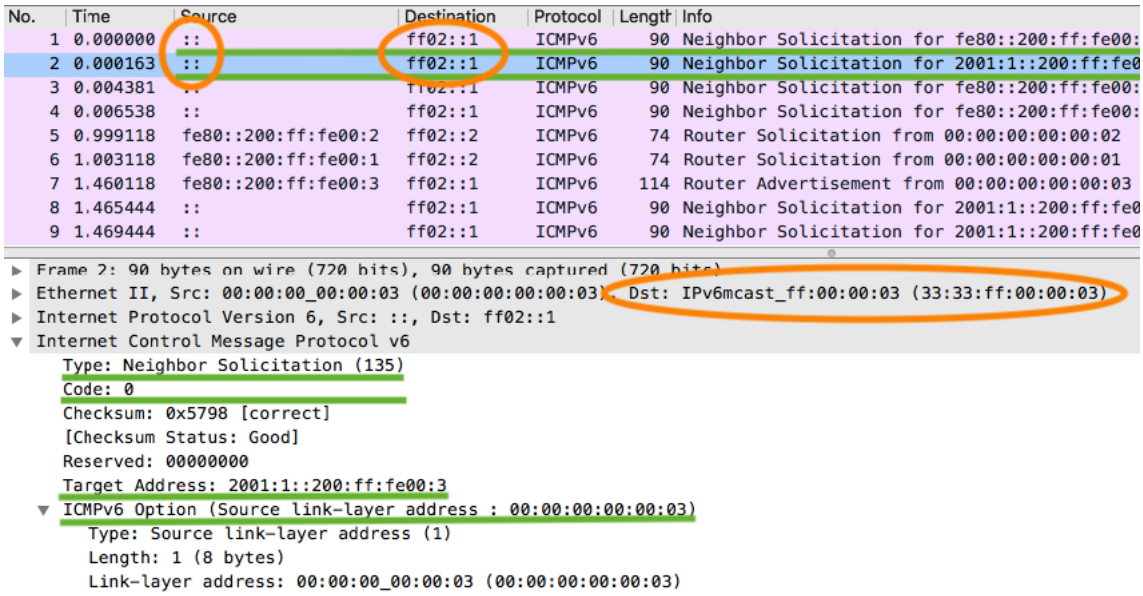
Statická konfigurace v tomto případě proběhla tak, že si po spuštění rozhraní směrovač `router1` nakonfiguroval link-local adresu (`fe80::200::ff:fe00:3`) a globální adresu (`2001::1::200:ff:fe00:3`) podle své MAC adresy (`00:00:00:00:00:03`). To je možné potvrdit v konzolovém výpisu 2.30 kde má směrovač `router1` (Node 2) již v čase 0 sekund na rozhraní 1 i 2 spolu s linkovými, uvedeny i globálně adresy. To, že směrovači byly přiděleny právě tyto adresy, musí nějak sdělit ostatním uzlům v síti, aby nevznikly případně kolize adres. K tomu slouží zprávy typu NS - Neighbor Solicitation a mechanismus DAD - Duplicate Address Detection.

Zpráva NS - Neighbor Solicitation a DAD - Duplicate Address Detection

V souboru `icmpv6-2-0.pcap` jsou vidět tato sdělení o přidělení adresy v podobně zpráv NS pro každou z adres směrovače `router1` - pakety č.1 - č.4.

Důležitými položkami zprávy NS, které lze z paketů zjistit jsou:

- *Typ* - zpráva NS má číslo 135.



Obr. 2.32: Obsah souboru icmpv6-2-0.pcap - pakety směrovače router1

- *Kód* - kód zprávy blíže definuje důvod vyvolání daného typu zprávy.
- *Cílová adresa (Target Address)* - adresa, kterou si uzel chce přidělit.
- *ICMPv6 volba (ICMPv6 Option)* - která je typu 1 a je v ní specifikována MAC adresa rozhraní, která tuto zprávu odesílá.

Všimněte si adresy odkud a kam jsou tyto zprávy zasílány. Zpráva NS je odeslána z adresy `::`, která značí zatím nespecifikovanou adresu a je posílána na adresu `ff02::1`.

Adresa `ff02::1` označuje multicastovou adresu skupiny pro všechny uzly v síti. Zpráva je na tuto adresu zasílána z důvodu, aby v případě, že danou adresu již nějaký z uzlů používá, mohl tento uzel na duplikátní přidělení zareagovat. V IPv6 je také používána multicastová adresa `ff02::2`, která obsahuje všechny směrovače dané sítě. Tyto adresy ve většině případů nahrazují broadcast, který v rámci protokolu IPv6 není. Tímto způsobem se zprávy ani nemusí přeposílat všem uzlům v síti ale jen těm, pro které je to nutné. Tabulka 2.5 popisuje shrnutí těchto adres.

IPv6 adresa	Účel/Dosah	IPv4 ekvivalent
ff02::1	Multicastová adresa skupiny všech uzlů / Jenom v lokální síti	Broadcast
ff02::2	Multicastová adresa skupiny všech smerovačů / Jenom v lokální síti	Broadcast

Tab. 2.5: Tabulka s popisem účelu a dosahu nejčastějších multicastových druhů IPv6 adres

Zprávou NS při přidělování si adresy vyvolává uzel mechanismus DAD - Duplicate Address Detection, který slouží k ověření unikátnost adresy v síti.

Proces DAD sestává z následujících kroků:

1. Při spuštění rozhraní se host přidává k link-local multicastové skupině pro všechny uzly - `ff02::1`.
2. Na multicastovou adresu skupiny všech uzlů je zaslaná zpráva NS s adresou, kterou si uzel chce přidělit.
3. Host po jistou dobu čeká, zda nějaký z uzlů nebude na zprávu NS reagovat - zprávou Neighbor Advertisement nebo další Neigbor Solicitation.
4. Pokud po dobu čekání nepřijde žádná odpověď, adresa se považuje za unikátní a rozhraní si ji může přiřadit.
5. Pokud nějaká odpověď přijde, adresa není unikátní. Host vyvolá konzolovou zprávu 'Duplicated address detected' a označí tuto adresu za nedostupnou.

Bezstavová konfigurace - SLAAC

V konzolovém výpisu 2.30 je vidět, že uzlům `host0`, `router0` a `host1` byly v době 0s přiděleny pouze link-local adresy. Globální adresy jim byly přiděleny až později, protože k jejich konfiguraci potřebovali informace obsažené ve zprávě *RA - Router Advertisement*.

Jak bylo uvedeno v kódu, směrovač `router1` neposílá automaticky do sítě 1 zprávu RA, ve které je specifikován prefix pro vytvoření adresy. Uzly `host0` a `router0` tak musí směrovač `router1` vyzvat k zaslání zprávy RA zprávou *RS - Router Solicitation*.

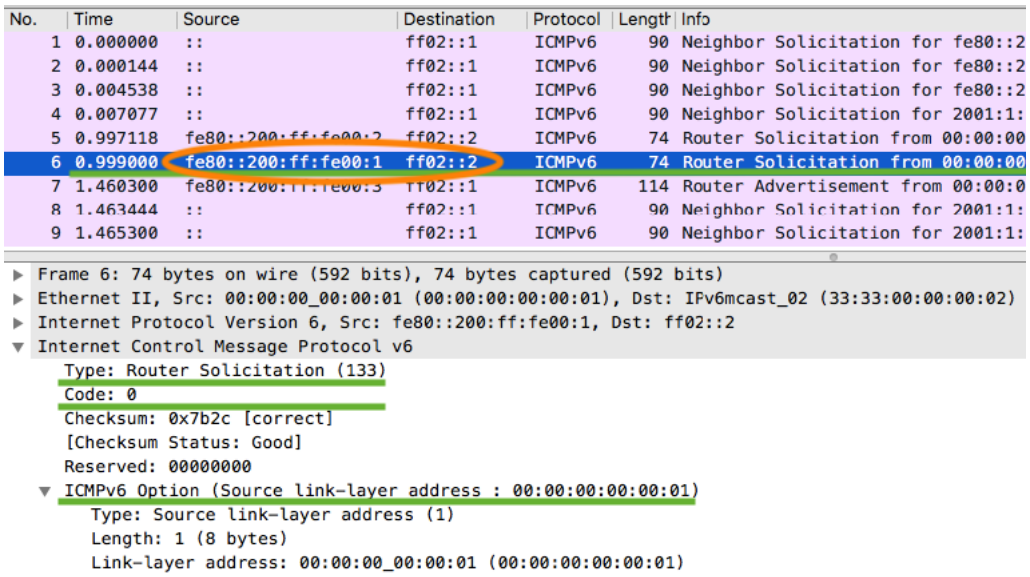
Otevřete si soubor `icmpv6-0-0.pcap`, který obsahuje pakety uzlu `host0`. Jeho obsah by měl shodovat s obrázkem 2.33 a sledujte jeho pakety.

V prvních paketech (č. 1 až 4.) je opět vidět NS zprávy pro DAD ověření a přidělení link-local adres uzlů host0, `router0` a `router1`. Dále je již však vidět jako uzly `host0` a `router0` posílají zprávu RS.

Zpráva RS - Router Solicitation

Jelikož uzly `host0` a `router0` nemají staticky nakonfigurovanou globální adresu, zasílají do sítě zprávu *RS - Router Solicitation*, kterou se snaží zjistit informace o směrovačích v jejich síti. Tyto zprávy je vidět v paketech č. 5 a 6. Zpráva RS má podobnou strukturu jako zpráva NS, obsahuje pole:

- *Typ* - zpráva RS má číslo 133.
- *Kód* - Kód č. 0, nemá pro tento typ žádnou další specifikaci.
- *ICMPv6 Volbu* - která je typu 1 a je v ní specifikována MAC adresa rozhraní, která tuto zprávu odesílá.



Obr. 2.33: Obsah souboru icmpv6-0-0.pcap - Detail zprávy RS, kterou zaslal uzel host0.

Zpráva RS, narozdíl od zprávy NS, už není zasílaná z nespecifikované adresy. Její zdrojovou adresou je link-local adresa uzlu `host0`, kterou si dosud věděl nakonfigurovat sám. Cílovou adresou je multicastové adresa skupiny všech směrovačů - `ff02::2`, protože uzel potřebuje získat informace od směrovače, ne všech uzlů. Na zprávu RS následně přichází od směrovače `router1` odpověď *RA - Router Advertisement*.

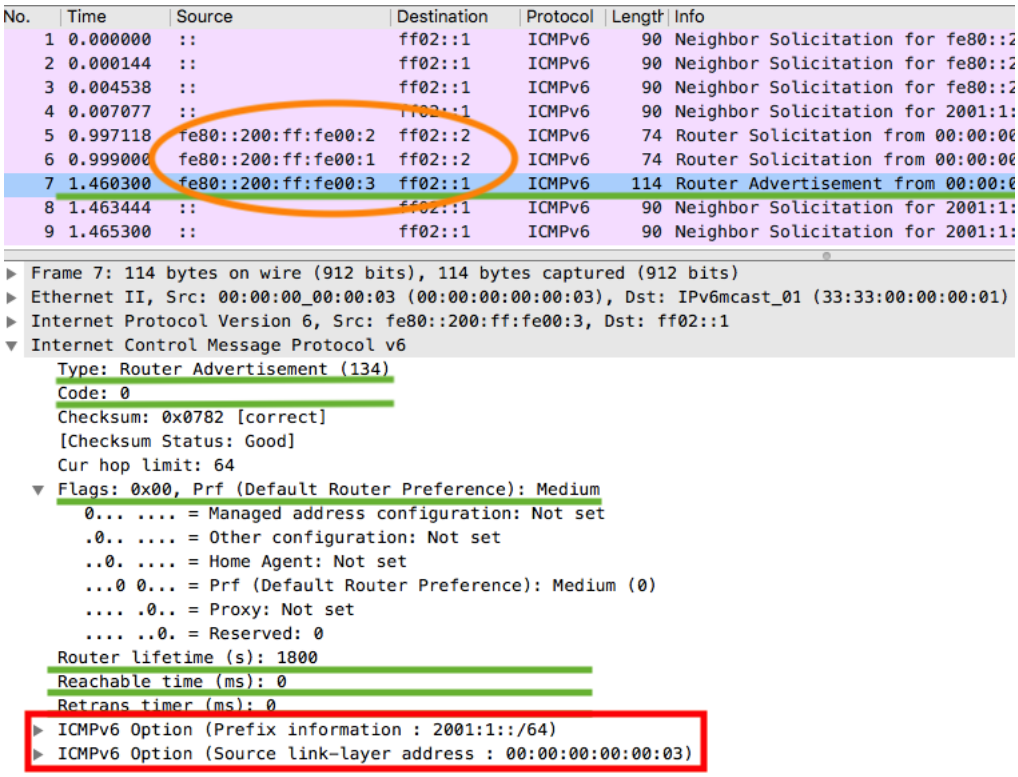
Zpráva RA - Router Advertisement

Obsah zprávy RA, kterou poskytl uzlu `host0` směrovač `router1` je zobrazen na obrázku 2.34. Zpráva RA má přiděleno číslo typu 134 a také defaultní kód 0. Co činí tuto zprávu důležitou, jsou nastavení jednotlivých příznaků. Hlavními příznaky pro konfiguraci globální adresy jsou M a O.

- *Flag M - Managed address configuration* - nastavení příznaku M znamená, že přidělování adres má na starosti DHCPv6 protokol.
- *Flag O - Other configuration flag* - nastavení příznaku O znamená, že je možné získat další konfigurační informace jako např. DNS server, pomocí protokolu DHCPv6.

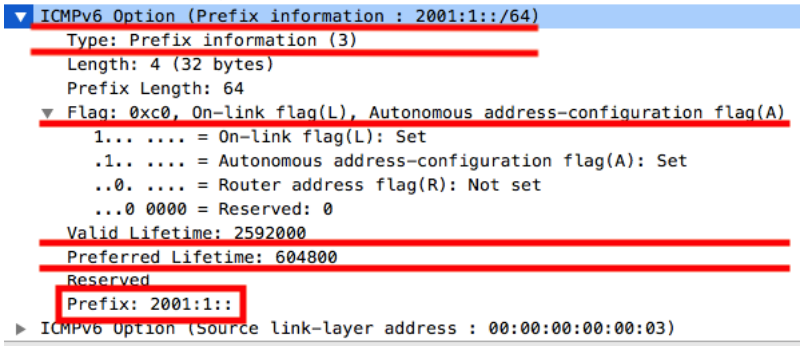
Ani jeden z příznaků by v rámci této ukázky nastavený být neměl. Uzly takto získávají informaci o tom, že v síti se pravděpodobně nenachází žádný aktivní DHCPv6 server.

Stavová konfigurace tedy není možná a globální adresu si uzel může nakonfigurovat bezstavově. Pro tuto konfiguraci potřebují uzly od směrovače `router1` prefix sítě, který by mohly použít. Pokud takový prefix existuje, bude uveden v poli *ICMPv6*



Obr. 2.34: Obsah souboru icmpv6-0-0.pcap - Detail zprávy RA pro uzly v síti od směrovače router1

Option s číslem typu 3 - *Prefix Information* 2.35. Toto pole obsahuje další příznaky



Obr. 2.35: Detail položky Prefix Information Option zprávy RA

upřesňující konfigurace adresy:

- *Flag L (On-link)* - nastavení příznaku L znamená, že uvedený prefix se nachází ve stejné síti jako daný uzel, jemuž byla RA zpráva doručena a může jej použít pro vytvoření adresy.
- *Flag A (Autonomous address-configuration)* - nastavení příznaku A znamená, že tento prefix může být použit pro bezstavový konfiguraci adresy.

Další související informace, které obsahuje část *ICMPv6 Option - Prefix Information* jsou:

- *Valid Lifetime* - je doba v sekundách, po kterou je prefix platný v dané síti pro možnost vytvoření adresy pro rozhraní.
- *Preferred Lifetime* - je doba v sekundách, po kterou zůstává adresa vygenerována bezstavovou konfigurací z daného prefixu preferovaná.
- *Prefix* - samotný prefix, který může být použit uzlem při konfiguraci adresy.

Podle těchto informací si ze získaného prefixu a své MAC adresy uzly dokážou za pomoci EUI-64 nakonfigurovat globální adresu pro své rozhraní. Nakonfigurovanou adresu však stále musí sdělit ostatním uzlům a zjistit tak, zda je v dané síti unikátní. To se děje opět pomoci zprávy NS, kterou je vidět jako poslední (paket č.9) na obrázku 2.34.

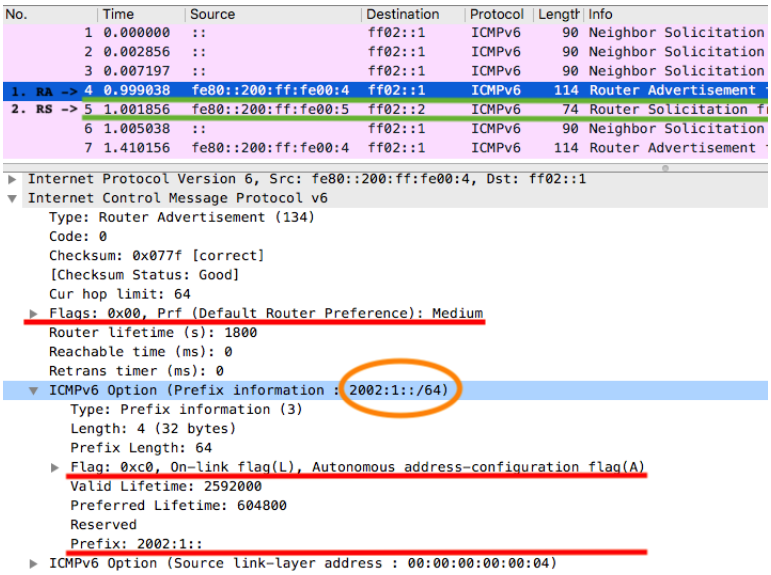
To, že uzel `host0` globální adresu získal až po vyžádání a získání informaci je vidět i v konzolovém výpisu. Ve výpisu 2.30 bylo vidět, že v čase 0s uzel `host0` (Node: 0) získal pouze link-local adresu, protože mu bylo přiděleno a spuštěno rozhraní. Obrázek 2.36 zobrazuje adresy uzlu v čase 2s, kde má toto rozhraní definovanou i globální adresu `2001:1::200:ff:fe00:1`, kterou si uzel vytvořil na základě získaného prefixu a své MAC adresy. Podobný proces absolvoval i uzel `router0` (Node: 1), který získal globální adresu také až po obdržení potřebných informací ve zprávě RA.

```
Node: 0 Time: 2s IPv6 addresses
(Interface index, Address index)        IPv6 Address
(0,0)   address: ::1/128; scope: HOST
(1,0)   address: fe80::200:ff:fe00:1/64; scope: LINK-LOCAL
(1,1)   address: 2001:1::200:ff:fe00:1/64; scope: GLOBAL

Node: 1 Time: 2s IPv6 addresses
(Interface index, Address index)        IPv6 Address
(0,0)   address: ::1/128; scope: HOST
(1,0)   address: fe80::200:ff:fe00:2/64; scope: LINK-LOCAL
(1,1)   address: 2001:1::200:ff:fe00:2/64; scope: GLOBAL
```

Obr. 2.36: Adresy přidělené uzlu host0 a směrovači router0 v čase 2s

SLAAC s automatickým Router Advertisment

Uzel `host1` v síti 2 měl rovněž nakonfigurovanou globální adresu pomocí bezstavové konfigurace SLAAC, s tím rozdílem, že do sítě 2 zasílá směrovač `router1` zprávu RA automaticky. Není tedy nutně potřeba, aby uzel `host1` nejprve zasílal výzvu RS. Otevřete soubor `icmpv6-3-0.pcap` pro uzel `host1`. V něm je vidět (obr. 2.37), že směrovač `router1` poslal do sítě 2 zprávu RA (paket č.4), v níž jsou všechny potřebné informace pro konfiguraci globální adresy - hlavně prefix sítě 2 - `2002:1::`. Uzel však zprávu RS (paket č.5) zaslal i tak, protože předem nevěděl, že směrovač `router1`



Obr. 2.37: Obsah souboru icmpv6-3-0.pcap - Automatické zaslání zprávy RA směrovačem router1 bez vyžádaní zprávou RS

zasílá zprávu RA automaticky. V konzolovém výpisu adres si dále ověřte, že uzel `host1` (Node: 3) získal globální adresu až později, po získání potřebných informací podobně jako uzly `host0` a `router0`.

Samostatná část A - DAD pro uzly se stejnou MAC adresou a EUI-64 metoda

Jak bylo zmíněno při statické konfiguraci adres 2.2.3, uzel pro přidělení adresy používá mechanismus *DAD - Duplicate Address Detection*, aby dal ostatním uzlům v síti informaci o adrese, kterou si chce na rozhraní přidělit. DAD pracuje za pomoci zprávy NS. Pokud na zprávu NS nějaký z uzlů zareaguje zprávou NA s tím, že tuto adresu používá on, adresa, kterou uzel oznamoval ve zprávě NS druhému uzlu nemůže být přidělena.

Tuto situaci je možné vyvolat například tak, že směrovači `router0` v síti 1, ručně změníme MAC adresu podle MAC adresy uzlu `host0` na `00:00:00:00:00:01`. Následující kód, který změní MAC adresu uzlu `router0`, vložte na místo v kódu označené komentářem *//Samostatna cast A - DAD pro stejnou MAC adresu:*. Při každém vkládání kódu dbejte, aby kopírovaný příkaz neobsahoval mezery, ty mohou způsobovat chyby při překladu programu

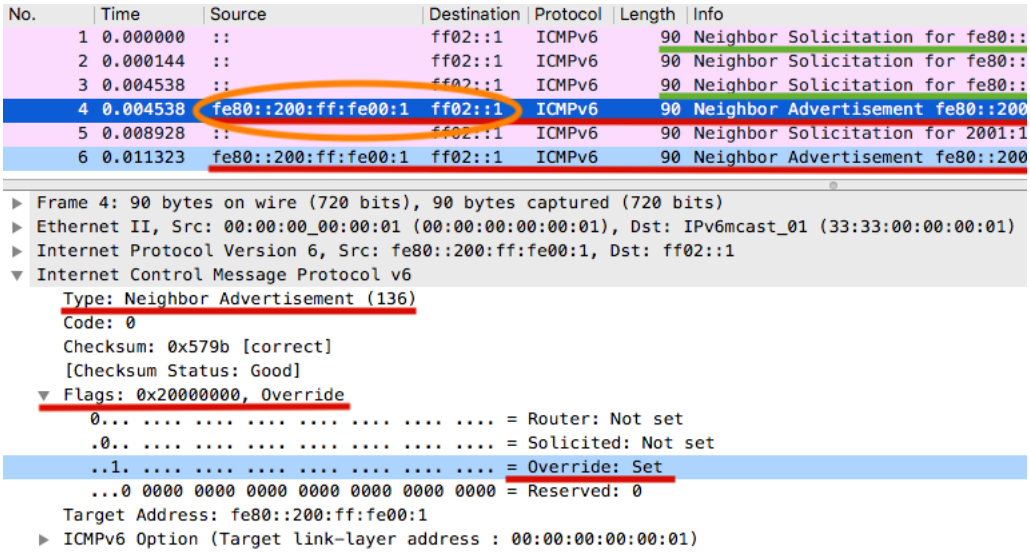
```
//Samostatna cast A - DAD pro stejnou MAC adresu:
 netDevices_net1.Get(1)->
    SetAddress(Mac48Address ("00:00:00:00:00:01"));
```

Spusťte si znovu simulaci.
Jakému uzlu se přidělila link-local adresa `fe80::200:ff:fe00:1`? Proč si uzly `host0` a `router0` podle konzolového výstupu (obr.2.38) ani později nedokázali přidělit globální adresu?

```
Node: 0 Time: 0s IPv6 addresses
(Interface index, Address index)        IPv6 Address
(0,0)   address: ::1/128; scope: HOST
(1,0)   address: fe80::200:ff:fe00:1/64; scope: LINK-LOCAL

Node: 1 Time: 0s IPv6 addresses
(Interface index, Address index)        IPv6 Address
(0,0)   address: ::1/128; scope: HOST
(1,0)   address: fe80::200:ff:fe00:1/64; scope: LINK-LOCAL

Node: 0 Time: 2s IPv6 addresses
(Interface index, Address index)        IPv6 Address
(0,0)   address: ::1/128; scope: HOST
(1,0)   address: fe80::200:ff:fe00:1/64; scope: LINK-LOCAL

Node: 1 Time: 2s IPv6 addresses
(Interface index, Address index)        IPv6 Address
(0,0)   address: ::1/128; scope: HOST
(1,0)   address: fe80::200:ff:fe00:1/64; scope: LINK-LOCAL
```

Obr. 2.38: DAD demonstrace - Adresy přidělené uzlům po přiřazení stejné MAC adresy jakou má uzel host0 uzlu router0

Otevřete si soubor s pakety pro uzel `host0` - `icmpv6-0-0.pcap`. Jeho obsah by měl shodovat s obrázkem 2.39.



Obr. 2.39: Obsah souboru icmpv6-0-0.pcap - Pakety uzlu host0 po přiřazení stejné MAC adresy jakou má uzel host0 uzlu router0

Na obrázku je vidět, že pakety č. 1 a č. 3 obsahují zprávu NS se stejnou hodnotou v poli *Target Address*, tedy se stejnou adresou, kterou si plánují přidělit a jsou zasílány ze stejné MAC adresy. Podle konzolového výpisu (obr.2.38) je vidět, že si oba tyto uzly link-local adresu `fe80::200:ff:fe00:1` i přidělili. Oba uzly tak reagují na zprávu druhého uzlu zprávou NA - Neighbor Advertisement s příznakem

Override, kde hlásí, že tuto adresu již používají. DAD tedy neproběhlo korektně a ani jeden z uzlů nemá potvrzenou svou link-local adresu.
Bez link-local adresy uzel nemůže komunikovat po síti a tudíž nemůže požádat směrovače v síti o zprávu RA vyvoláním zprávy RS. Bez link-local adresy, si uzel nemůže požádat ani o globální adresu a je nutný zásah administrátora, který by jednomu z uzlů v síti změnil MAC adresu pro tuto síť.
Zprávy typu NS/ NA - Neighbor Solicitation / Neighbor Advertisement nejsou jen součástí mechanismu DAD, jejich další funkcionalitou je získání MAC adresy podle přidělené IP adresy. To bude blíže specifikováno později. Pro další pokračování v úkolu, zakomentujte příkaz pro změnu MAC adresy směrovači `router0` //netDevices_net1.Get(1)->SetAddress(Mac48Address ("00:00:00:00:00:01"));.

Samostatná část B - Metóda EUI-64

Adresa pro rozhraní je v rámci simulátoru NS3 stále vytvářena za pomoci algoritmu EUI-64. Zjistěte jak funguje tento algoritmus a zobrazte si pomocí následující části kódu, jaká IPv6 adresa by byla přidělena pro MAC adresu - `12:34:56:AB:CD:E7`. Kód zkopírujte na místo označené komentářem *//Samostatna cast B - Metoda EUI-64:*. Při každém vkládání kódu dbejte na to, aby kopírovaný příkaz neobsahoval mezery, ty mohou způsobovat chyby při překladu programu.

```
Mac48Address special_mac = ("12:34:56:AB:CD:E7");
Ipv6Address special_prefix = ("2002:2::");
std::cout << "IPv6␣adresa␣je:␣" <<
Ipv6Address::MakeAutoconfiguredAddress
    (special_mac, special_prefix) << "\n" << std::endl;
```

Jak by se z výsledku dala zpětně získat MAC adresa? Zjistěte jak si uzly v rámci IPv6 dokáží zjistit MAC adresu za pomoci IP adres. Mohou se spoléhat pouze na zpětnou transformaci podle EUI-64? Odpověď na tuto otázku je - *ne*, nemůžou. Protože ne každý uzel v reálných sítích musí použít pro vytvoření své adresy algoritmus EUI-64. Zjišťování MAC adres dle příslušných IPv6 adres má také v režii protokol ICMPv6. To jak přesně probíhá, bude popsáno v části 2.2.4.

Samostatná část C - RA s více prefixy

V bezstavové konfiguraci může zpráva RA - Router Advertisement obsahovat i několik prefixů, které budou obsaženy v položkách Prefix Information Option. Jak zareaguje uzel jestliže při bezstavové konfiguraci dostane zprávu RA s více prefixy, jaký z nich si vybere pro konfiguraci adresy?

Přidejte následující řádek kódu na místo v kódu označené komentářem *//Samostatna cast C - RA s vice prefixy:*. Tento dodatek způsobí, že směrovač `router1` nebude do sítě 1 zasílat jen prefix `2001:1::`, ale i prefix `2002:1::`.

```
//Samostatna cast C - RA s vice prefixy:
radvdHelper.AddAnnouncedPrefix(
    interface_router1_net1.GetInterfaceIndex(0),
    net2_address, 64);
```

Spusťte znovu simulaci. V konzolovém výpisu se podívejte s jakým prefixem byly uzlům `host0` a `router0` přidělené globální adresy. Konzolový výpis by měl být shodný s výpisem na obrázku 2.40. Uzel si nemusí vybírat, který z prefixů pro rozhraní použije. Protokol IPv6 podporuje funkcionalitu multihomingu a tedy jedno rozhraní může být součástí více sítí. To se děje například v případě, že chtěl mít uživatel několika ISP, přičemž každý ISP by mu poskytl svůj prefix.

```
Node: 0 Time: 2s IPv6 addresses
(Interface index, Address index)   IPv6 Address
(0,0)  address: ::1/128; scope: HOST
(1,0)  address: fe80::200:ff:fe00:1/64; scope: LINK-LOCAL
(1,1)  address: 2002:1::200:ff:fe00:1/64; scope: GLOBAL
(1,2)  address: 2001:1::200:ff:fe00:1/64; scope: GLOBAL

Node: 1 Time: 2s IPv6 addresses
(Interface index, Address index)   IPv6 Address
(0,0)  address: ::1/128; scope: HOST
(1,0)  address: fe80::200:ff:fe00:2/64; scope: LINK-LOCAL
(1,1)  address: 2002:1::200:ff:fe00:2/64; scope: GLOBAL
(1,2)  address: 2001:1::200:ff:fe00:2/64; scope: GLOBAL
```

Obr. 2.40: Adresy přidělené uzlům po získání zprávy RA s více prefixy

Pro další pokračování v úkolu zakomentujte nebo vymažte řádek s přidáním dalšího prefixu pro síť 1 *//radvdHelper.AddAnnouncedPrefix(...)*.

2.2.4 2. část: Směrování

Další část úkolu bude věnována zprávě ICMPv6 Redirect, která pomáhá uzlům zasílat své pakety efektivnější cestou, pokud je to v dané síti možné. V této části bude také vysvětleno jakým způsobem zastává protokol ICMPv6 funkcionalitu protokolu ARP, který slouží pro získání MAC adresy dle IP adresy. V závěru této části bude zmíněný protokol RIPng, který slouží k výměně směrovacích informací v IPv6 sítích.

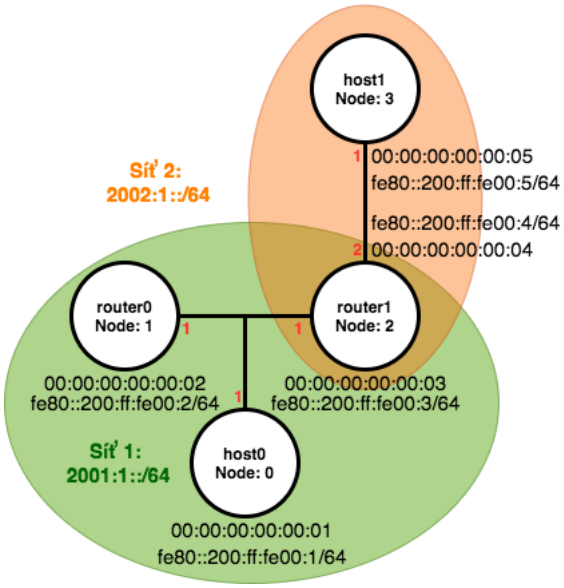
Statické směrování

Jedním ze způsobů nastavení směrování v síti je statické směrování. Na začátek je pro účely této úlohy potřebné pro všechny uzly definovat statickou konfiguraci IPv6 adres. Vymažte celou část konfigurace adres, od příkazu `ipv6.SetBase(net1_address,`

`Ipv6Prefix(64));` po příkaz `radvdApps.Stop (Seconds(10.0));` a nahraďte ji následujícím kódem, který nakonfiguruje globální adresy všem uzlům staticky. Při každém vkládání kódu dbejte aby kopírovný příkaz neobsahoval mezery, ty mohou způsobovat chyby při překladu programu.

```
ipv6.SetBase (net1_address, Ipv6Prefix (64));
Ipv6InterfaceContainer interfaces_net1 =
    ipv6.Assign (netDevices_net1);
interfaces_net1.SetForwarding (2, true);
interfaces_net1.SetForwarding (1, true);

ipv6.SetBase (net2_address, Ipv6Prefix (64));
Ipv6InterfaceContainer interfaces_net2 =
    ipv6.Assign(netDevices_net2);
interfaces_net2.SetForwarding (0, true);
```

Tato část úkolu bude demonstrovat, že protokol ICMPv6 má možnost změnit trasu provozu pokud tato trasa není nejlepší trasou v dané síti. Při ponechání bezstavové konfigurace pro uzel `host0`, by uzel považoval `router1` za svou výchozí cestu a tudíž by svůj provoz rovnou posílal na něj, což by byla nejlepší cesta v dané topologii. Proto budou všem uzlům adresy přiděleny staticky a následující kód pro ně definuje statické cesty, které *nebudou udávat nejlepší cestu* pro pakety směrovány od uzlu `host0` k uzlu `host1`. Hodnoty přidělených adres se nezmění, protože jsou generovány pomocí EUI-64 a topologie tak zůstává stejná, jako je znázorněna na obrázku 2.41.



Obr. 2.41: Topologie s adresami uzlů

Přidělení statických cest vložte za příkaz *interfaces_net2.SetForwarding (0, true);*.

```
interfaces_net1.SetDefaultRoute(0, 1);
interfaces_net1.SetDefaultRoute(1, 2);
interfaces_net2.SetDefaultRouteInAllNodes(0);
```

Pomocí tohoto kódu si uzel `host0` (v proměnné `interfaces_net1` má tento uzel index 0) nastaví pro svou výchozí cestu adresu směrovače `router0` (v proměnné `interfaces_net1` má tento uzel index 1) a směrovači `router0` bude nastavena adresa pro výchozí cestu na adresu směrovače `router1` (v proměnné `interfaces_net1` má tento uzel index 2). V síti 2 bude pro oba uzly výchozím směrovačem uzel `router1` (v proměnné `interfaces_net2` má tento uzel index 0).
Tímto je docílena následující cesta zasílání zprávy od uzlu `host0` k uzlu `host1`:

- jako první uzel `host0` zasílá zprávu na adresu své výchozí cesty - směrovači `router0`.
- směrovač `router0` zasílá zprávu dále na adresu své výchozí cesty - směrovači `router1`.
- směrovač `router1` má danou síť přímo připojenou a zprávu dále přepošle uzlu `host1`.

Tato cesta není v této topologii nejlepší, jelikož uzel `host0` by mohl přímo zaslat zprávu směrovači `router1`. To, že tato cesta není nejkratší protokol ICMPv6 zjistí po odeslání první zprávy *Echo Request* od uzlu `host0` a směrovač `router0` zašle uzlu `host0` zprávu ICMPv6 Redirect, v níž uzlu `host0` sdělí, aby další pakety pro uzel `host1` zasílal rovnou na adresu směrovače `router1`. Tím se budou zprávy zasílat nejlepší cestou i bez nutnosti měnit dané statické cesty administrátorem.

Aplikace Ping

Pro zasílání zpráv bude v simulaci sloužit aplikace Ping, která používá zprávy *ICMPv6 Echo Request* a *Echo Reply*. Detaily těchto zpráv budou popsány později. Zdrojovou adresou pro zprávu Echo Request bude adresa uzlu `host0` - `2001:1::200:ff:fe00:1` a cílovou adresou je adresa uzlu `host1` - `2002:1::200:ff:fe00:5`. Zaslaných bude 5 zpráv, vždy sekundu po sobě a velikost jedné zprávy bude 1024B. Aplikace je přidělena uzlu `host0` a zprávy začne zasílat v čase 2s. Následující kód vložte za definování poslední statické cesty pro síť 2 za příkaz interfaces_net2.SetDefaultRouteInAllNodes(0);. Při každém vkládání kódu dbejte aby kopírovaný příkaz neobsahoval mezery, ty mohou způsobovat chyby při překladu programu

```
Ping6Helper ping6;
ping6.SetLocal (Ipv6Address("2001:1::200:ff:fe00:1"));
ping6.SetRemote (Ipv6Address("2002:1::200:ff:fe00:5"));
```

```
ping6.SetAttribute ("MaxPackets", UintegerValue (5));
ping6.SetAttribute ("Interval", TimeValue (Seconds (1.)));
ping6.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer apps = ping6.Install (host0);
apps.Start (Seconds (2.0));
apps.Stop (Seconds (10.0));
```

V této části bude důležitá změna ve směrovací tabulce uzlu `host0`. Tabulka bude vypsána nejprve v čase 2s, kdy došlo k zaslání první Echo Request zprávy a kdy bude zpráva ještě směrována neoptimalizovanou cestou na směrovač `router0`. Dále bude tabulka vypsána v době 3s, kdy už uzlu `host0` přišla informace o existenci lepší cesty (zprávou Redirect) pro tento provoz a uzel si vytvořil nový záznam, podle kterého už další zprávy Echo Request směřuje na směrovač `router1`. Následující kód vypíše směrovací tabulku uzlu `host0` v čase 2 a 3s, vložte jej za příkaz získání PCAP souborů - csma.EnablePcapAll (ĬCMPv6");

```
Ipv6StaticRoutingHelper routingHelper;
Ptr<OutputStreamWrapper> routingStream =
    Create<OutputStreamWrapper> (&std::cout);
routingHelper.PrintRoutingTableAt (Seconds (2.0), host0,
    routingStream);

routingHelper.PrintRoutingTableAt (Seconds (3.0), host0,
    routingStream);
```

Výpis přehledu přidělených adres pro jednotlivé uzly můžete v kódu nechat, pro lepší přehlednost je ale vhodné jej vymazat nebo zakomentovat. Po doplnění kódu spusťte simulaci znovu. Po jejím dokončení by se měl výpis v konzoli shodovat s obrázkem 2.42.

```
Node: 0, Time: +2.0s, Local time: +2.0s, Ipv6StaticRouting table
Destination                 Next Hop                  Flag Met Ref Use If
::1/128                     ::                        UH   0   -   -   0
fe80::/64                   ::                        U    0   -   -   1
2001:1::/64                 ::                        U    0   -   -   1
::/0                        fe80::200:ff:fe00:2       UG   0   -   -   1

Node: 0, Time: +3.0s, Local time: +3.0s, Ipv6StaticRouting table
Destination                 Next Hop                  Flag Met Ref Use If
::1/128                     ::                        UH   0   -   -   0
fe80::/64                   ::                        U    0   -   -   1
2001:1::/64                 ::                        U    0   -   -   1
::/0                        fe80::200:ff:fe00:2       UG   0   -   -   1
2002:1::200:ff:fe00:5/128   fe80::200:ff:fe00:3       UH   0   -   -   1
```

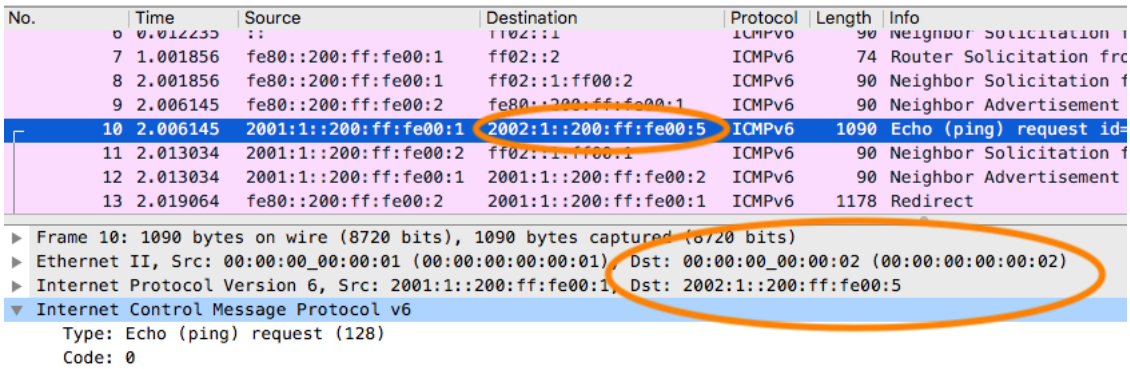
Obr. 2.42: Směrovací tabulka uzlu host0 v čase 0s a 3s

Z konzolového výpisu je vidět, že během simulace si uzel změnil svou směrovací tabulku tím, že si do ní přidal nový záznam pro uzel `2002:1::200:ff:fe00:5`.

Tuto změnu způsobila zpráva ICMPv6 Redirect. První zpráva *Echo Request* se řídila směrovací tabulkou uzlu `host0` v čase 2s, kdy byl veškerý neznámý provoz z uzlu `host0` směrován na jeho výchozí směrovač `router0` (řádek: `::/0 -> fe80::200::ff:fe00:2`). Po obdržení zprávy *Redirect*, si uzel `host0` změnil svou směrovací tabulku přidáním záznamu o směrování provozu pro uzel `host1` (řádek: `2002:1::200:ff:fe00:5 -> fe80::200::ff:fe00:3`) a tedy v době 3s už uzel `host0` věděl, že zprávu *Echo Request* pro uzel `host1` bude lépe zaslat na adresu směrovače `router1`. Otevřete si soubor s pakety pro uzel `host0 icmpv6-0-0.pcap` a najděte paket se zprávou *ICMPv6 Redirect*.

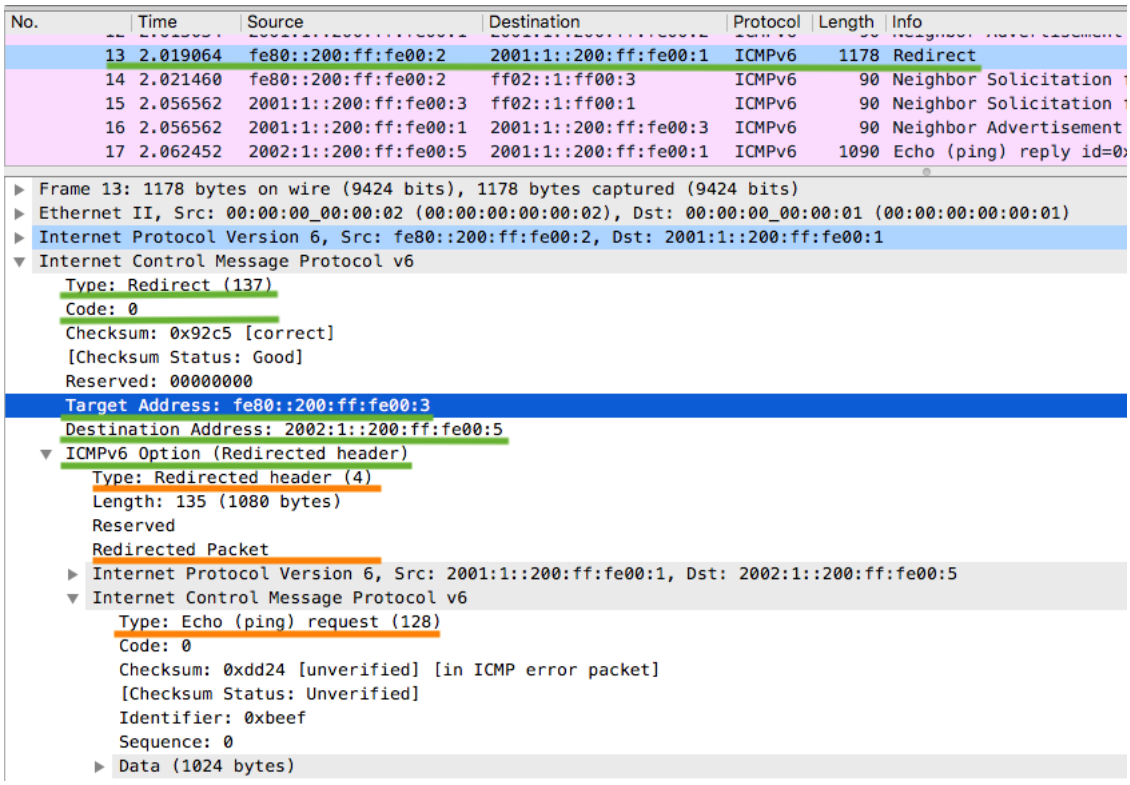
Zpráva Redirect

Tato zpráva je vyvolána, pokud se v síti nachází více směrovačů a některý z nich zjistí, že pro provoz na danou cílovou adresu od daného uzlu existuje efektivnější cesta přes jiný směrovač, než jaký používal dosud. Tedy pokud směrovač zjistí, že daný uzel nemá pro danou cílovou adresu efektivní next-hop adresu.
To se děje i v tomto případě, protože výchozí cestou pro uzel `host0` byla adresa směrovače `router0`. Jeho obsah by měl shodovat s obrázkem 2.43. Najděte paket s první zprávou Echo request (na obrázku paket č. 10). Uzel `host0` odesílá tuto zprávu ze své adresy na globální adresu uzlu `host1`, jako cílovou MAC adresu uvádí MAC adresu směrovače `router0`, protože to je jeho next-hop směrovač podle výchozí cesty ve směrovací tabulce . Po přijetí zprávy Echo Request směrovač `router0` zjistí, že



Obr. 2.43: Obsah souboru icmpv6-0-0.pcap - Detail zprávy Echo Request na cílovou MAC adresu zprávy

on bude tuto zprávu zasílat na směrovač `router1`. Zároveň ví, že směrovač `router1` se nachází stejné síti jako i uzel `host0`, od něhož tato zpráva přišla. Zasílá tedy uzlu `host0` zprávu *Redirect* 2.44, aby si uzel `host0` pro provoz na danou cílovou adresu změnil svou směrovací tabulku a zasílal tento provoz rovnou na směrovač `router1`, protože jsou ve stejné síti a tato cesta je efektivnější. To vše je uvedeno

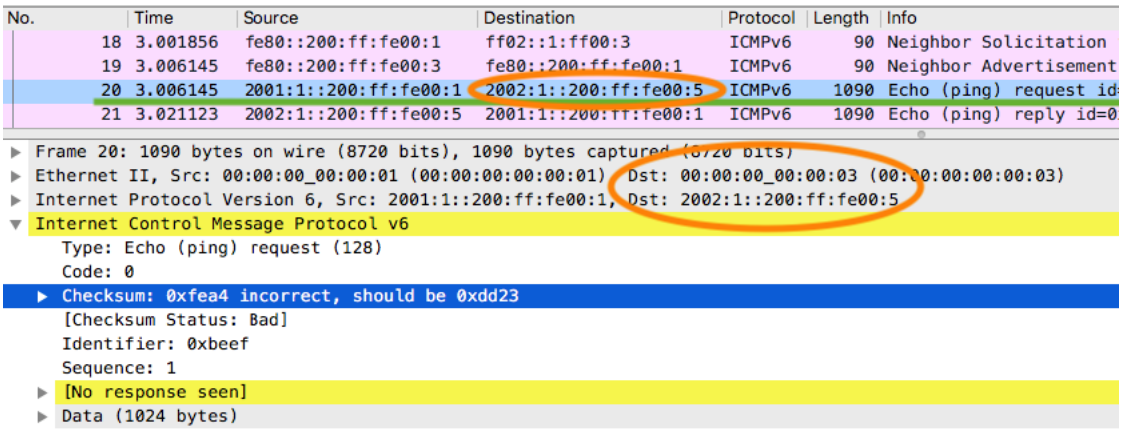


Obr. 2.44: Obsah souboru icmpv6-0-0.pcap - Detail zprávy Redirect odeslané z uzlu router0

ve zprávě *Redirect*. V souboru `icmpv6-0-0.pcap` zprávu najděte a rozbalte si její detaily. Zpráva by měla shodovat se zprávou na obrázku 2.44. Zpráva Redirect má podobnou strukturu jako předešlé ICMPv6 zprávy, obsahuje:

- *Typ* - pro zprávu Redirect je definováno číslo typu 137.
- *Kód* - má definovaný pouze kód 0, který neobsahuje žádné další specifikace zprávy.
- *Target Address* - v tomto poli zpráva definuje na jakou adresu si má daný uzel změnit ve směrovací tabulce next-hop adresu pro provoz s cílovou adresou uvedenou v následujícím poli.
- *Destination Address* - toto pole obsahuje adresu cílového provozu, pro který se uzlu mění next-hop adresa.
- *ICMPv6 Option* - pro tuto zprávu, je definován ICMPv6 Option s typovým číslem 4 - *Redirected header*, který specifikuje jaký provoz způsobil tuto změnu. V tomto případě to byla zpráva Echo Request protokolu ICMPv6.

Najděte a zobrazte si druhou zprávu Echo Request (na obrázku 2.45 paket č. 20), kterou uzel `host0` zaslal. Další zprávu Echo Request již uzel `host0` zasílá stále na globální adresu uzlu `host1`, jako cílovou MAC adresu již uvádí adresu směrovače `router1`, protože pro tento provoz je jeho next-hop adresou již směrovač `router1`.



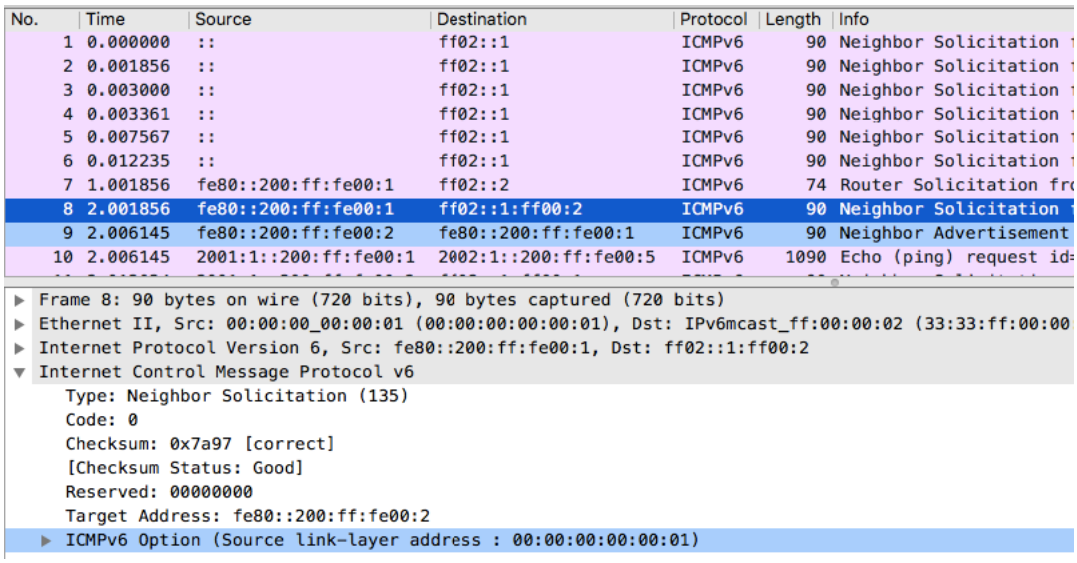
Obr. 2.45: Obsah souboru icmpv6-0-0.pcap - Detail v poradí druhé odeslané zprávy Echo Request z uzlu host0 se změněnou cílovou MAC adresou

To potvrzuje i směrovací tabulka uzlu `host0` v konzolovém výpisu, která si později přidala záznam pro uzel `host1` (obr.2.42).

Zprávy NS/NA jako ARP pro ICMPv6

V tomto souboru je také možné vidět další funkcionalitu protokolu ICMPv6 protokolu - *zjištění MAC adresy uzlu z jeho IP adresy.*
Aby uzel `host0` mohl doručit zprávu Echo request uzlu `host1`, potřebuje zjistit jeho MAC adresu. Tuto funkci má v rámci IPv4 na starosti protokol ARP. Pro protokol IPv6 zajišťuje tuto funkcionalitu protokol ICMPv6 a využívá k tomu opět dvojici zpráv *NS/NA* podobně jako u DAD.
V souboru `icmpv6-0-0.pcap` najděte pakety zpráv *NS/NA*, které uzel zasílá před odesláním zprávy Echo request (na obrázku 2.46 to jsou pakety č.8 a 9). Zprávu NS uzel `host0` zasílá, protože potřebuje do paketu se zprávou Echo request doplnit nejen IPv6 adresu, ale i MAC adresu uzlu, kterému ji má odeslat. Uzel `host0` má sice definováno, že zpráva Echo request má být poslána na adresu `2002:1::200:ff:fe00:5`, ale protože uzel nemá žádnou informaci o tom, kde se síť s prefixem `2002:1::` nachází, potřebuje zadat na místo *DST MAC address* ve zprávě Echo request, MAC adresu svého výchozího směrovače `router0`.

O jeho MAC adresu žádá uzel `host0` zasláním zprávy *NS* (paket č.8). Zdrojovou adresou této zprávy je link-local adresa uzlu `host0` a cílovou adresou je další typ multicastové adresy - adresa *solicited-node.*
Solicited-node adresa je další multicastovou adresou, vytvoření které je od uzlu vyžadováno. Tato adresa vznikne přidáním posledních 24 bitů z unicastové adresy k prefixu `ff02:0:0:0:0:1:ff00::/104`. Jelikož je vyžadováno si tento druh adresy vytvořit pro každou svou unicastovou adresu a každý uzel si ji umí odvodit pro



Obr. 2.46: Obsah souboru icmpv6-0-0.pcap - Detail zprávy NS pro získání MAC adresy směrovače router0

libovolný uzel v síti, je *solicited-node* adresa využívaná při zjišťování MAC adresy uzlu.

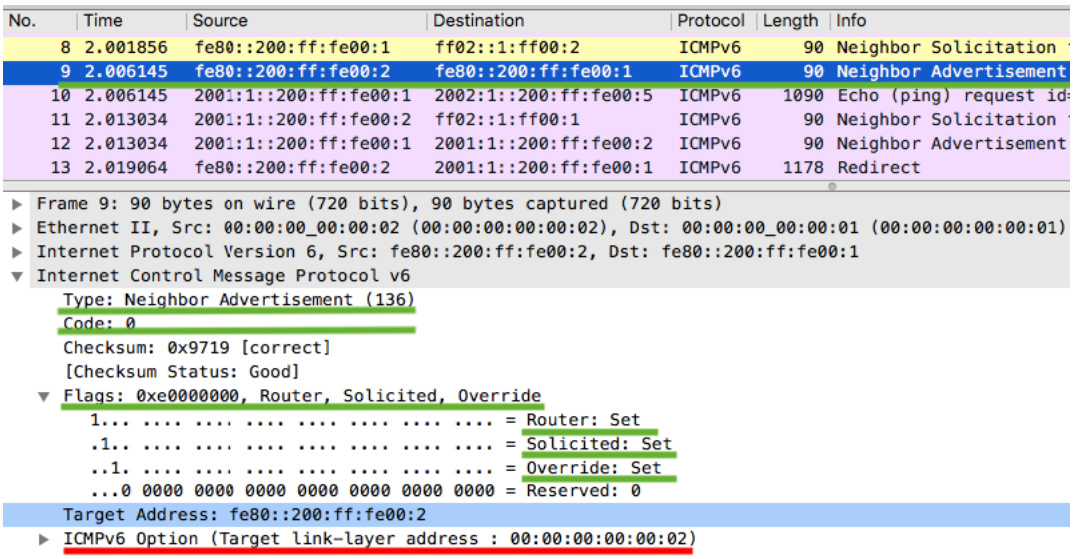
Právě zde spočívá rozdíl vůči protokolu ARP, který při zjišťování MAC adresy uzlu zasílá dotaz na tuto adresu každému uzlu v síti broadcastem. Pro protokol IPv6 však broadcast definován není a používa se multicast.

Uzel `host0` zasílá zprávu NS na multicastovou *solicited-node* adresu pro směrovač `router0`, kterou si uzel `host0` dokáže vypočítat z jeho link-local adresy (`fe80:200::ff:fe00:2`). K *solicited-node* adrese uzlu `router0` si také umí uzel `host0` vypočítat její příslušnou MAC adresu - `33:33:ff:00:00:02` - viz. obr. 2.46. Tímto při zjišťování MAC adresy uzel `host0` nemusí zatěžovat dotazem každý uzel v síti, jako to dělá protokol ARP, ale přesně specifikuje dotaz pouze pro rozhraní uzlu, kterého se týká.

Zpráva NA jako ARP pro ICMPv6

Odpovědí na zprávu NS je zpráva NA - Neighbor Advertisement. Zobrazte si detaily zprávy NA, která je odpovědí na zprávu NS pro získání MAC adresy uzlu `router0`. Její obsah by se měl shodovat s obrázkem 2.47. Tato zpráva je zasílána přímo od uzlu, jehož MAC adresu bylo žádané zjistit a jejím cílem je uzel, který o MAC adresu žádal. Struktura této zprávy zůstává podobná zbylým ICMPv6 zprávám. Zpráva obsahuje:

- *Typ* - pro zprávu NA je přiděleno číslo typu 136.
- *Kód* - zpráva neobsahuje žádný detailnější popis, proto je číslo kódu 0.



Obr. 2.47: Obsah souboru icmpv6-0-0.pcap - Detail zprávy NA pro získání MAC adresy směrovače router0

- *Flags - Příznaky*
  - *R - Router* - je nastaven, pokud je odesílatelem směrovač.
  - *S - Solicited* - je nastaven, pokud se jedná o odpověď na zprávu NS.
  - *O - Override* - je nastaven, pokud má tato zpráva přepsat existující data záznamu a aktualizovat link-local adresu.
- *Target Address* - zde je adresa uzlu, pro který byla zjišťována MAC adresa.
- *ICMPv6 Option* - toto pole obsahuje MAC adresu uzlu, jehož IP adresa se nachází v poli *Target Address*.

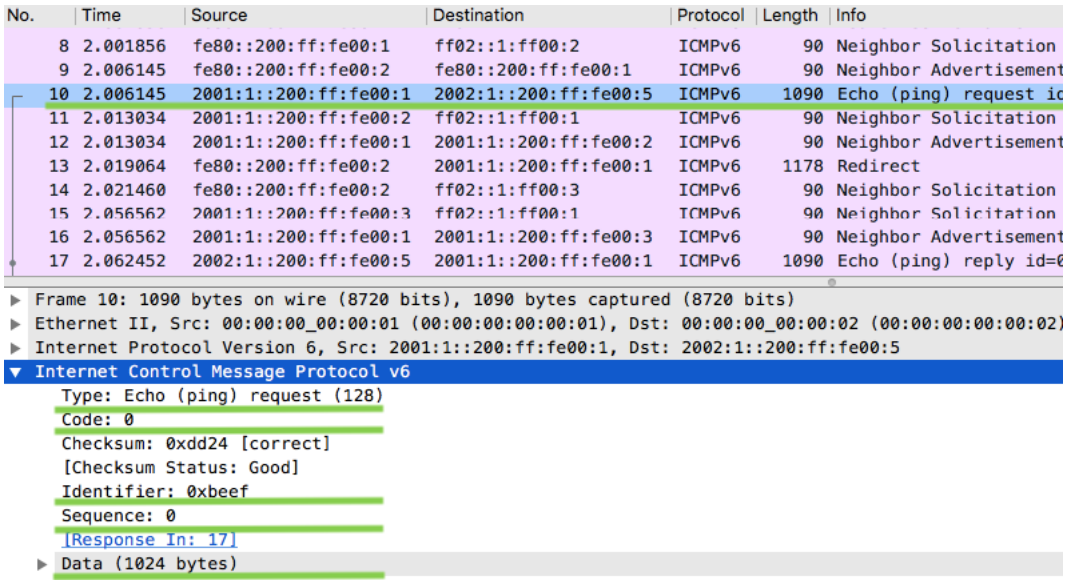
Po této odpovědi ví uzel `host0` doplnit i MAC adresu cíle pro zprávu Echo Request, kterou potřeboval.

Zprávy Echo Request a Echo Reply

Aplikace *Ping* je používána pro ověření dostupnosti spojení mezi dvěma síťovými rozhraními. I v rámci IPv4 i pro IPv6, je na tuto funkci využíván protokol ICMP, přesněji jeho zprávy *Echo Request* a *Echo Reply*. Detailnější výpis zprávy *Echo Request* je možné získat z paketu, který nese její informace. Tyto pakety si najděte a podívejte se na jejich obsah, ty by měly shodovat s obrázky 2.48 pro zprávu Echo Request a 2.49 pro zprávu Echo Reply.

Zpráva obsahuje položky:

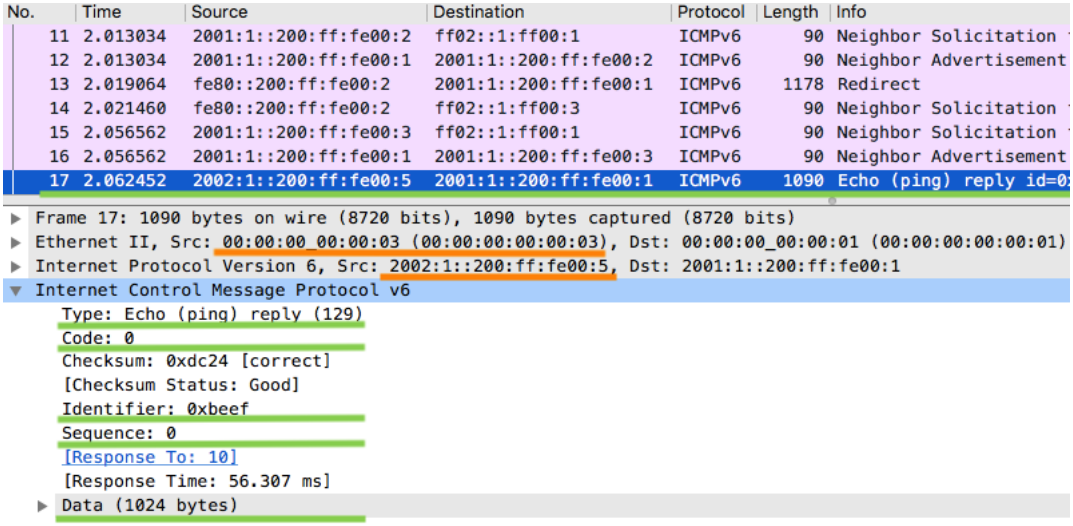
- *Typ* - číslo typu, pro zprávu Echo Request je 128
- *Kód* - má definovaný výchozí kód 0, který neobsahuje žádné další detaily
- *Identifier* - identifikátor, který pomáhá při přiřazení odpovědi k této žádosti.



Obr. 2.48: Obsah souboru icmpv6-0-0.pcap - Detail zprávy Echo Request

- *Sequence* - pořadové číslo, které také pomáhá při přiřazení odpovědi k této žádosti.
- *Data* - nula nebo více oktetů libovolných dat. V tomto případě byla velikost paketů definována na 1024B.

Pokud je spojení funkční a druhý uzel je dostupný, odpovídá druhý uzel zprávou *Echo Reply*, která je zobrazena na obrázku 2.49 a obsahuje položky:



Obr. 2.49: Obsah souboru icmpv6-0-0.pcap - Detail zprávy Echo Reply

- *Typ* - číslo typu, pro zprávu Echo Reply je to 129.
- *Kód* - má definovaný defaultní kód 0, který neobsahuje žádné další detaily.

- *Identifier* - identifikátor, který pomáhá při přiřazení odpovědi k žádosti. Měl by se shodovat s identifikátorem v žádosti.
- *Sequence* - pořadové číslo, které pomáhá pomáhá při přiřazení odpovědi k žádosti.
- *Data* - nula nebo více oktetů libovolných dat. V tomto případě byla velikost paketů definována na 1024B.

Z tohoto paketu je také vidět, že zdrojovou adresou je adresa uzlu `host1`, ale odpověď přichází ze směrovače `router1`, jelikož zdrojovou MAC adresou je MAC adresa směrovače `router1`.

Shrnutí komunikace pomocí informačních zpráv protokolu ICMPv6

Postupně si prohlížejte pakety uzlu `host0` v souboru `icmpv6-0-0.pcap` podle obrázku 2.50, kde jsou barevně rozlišeny různé části komunikace, kterou má plně v režii protokol ICMPv6. Komunikace probíhá následovně:

No.	Time	Source	Destination	Protocol	Length	Info
1. DAD	1 0.000000	::	ff02::1	ICMPv6	90	Neighbor Solicitation
	2 0.001856	::	ff02::1	ICMPv6	90	Neighbor Solicitation
	3 0.003000	::	ff02::1	ICMPv6	90	Neighbor Solicitation
	4 0.003361	::	ff02::1	ICMPv6	90	Neighbor Solicitation
	5 0.007567	::	ff02::1	ICMPv6	90	Neighbor Solicitation
	6 0.012235	::	ff02::1	ICMPv6	90	Neighbor Solicitation
2. host0 RS	7 1.001856	fe80::200:ff:fe00:1	ff02::2	ICMPv6	74	Router Solicitation fr
3. ARP	8 2.001856	fe80::200:ff:fe00:1	ff02::1:ff00:2	ICMPv6	90	Neighbor Solicitation
	9 2.006145	fe80::200:ff:fe00:2	fe80::200:ff:fe00:1	ICMPv6	90	Neighbor Advertisement
4. Request	10 2.006145	2001:1::200:ff:fe00:1	2002:1::200:ff:fe00:5	ICMPv6	1090	Echo (ping) request id=
5. ARP	11 2.013034	2001:1::200:ff:fe00:2	ff02::1:ff00:1	ICMPv6	90	Neighbor Solicitation
	12 2.013034	2001:1::200:ff:fe00:1	2001:1::200:ff:fe00:2	ICMPv6	90	Neighbor Advertisement
6. Redirect	13 2.019064	fe80::200:ff:fe00:2	2001:1::200:ff:fe00:1	ICMPv6	1178	Redirect
7. ARP	14 2.021460	fe80::200:ff:fe00:2	ff02::1:ff00:3	ICMPv6	90	Neighbor Solicitation
	15 2.056562	2001:1::200:ff:fe00:3	ff02::1:ff00:1	ICMPv6	90	Neighbor Solicitation
	16 2.056562	2001:1::200:ff:fe00:1	2001:1::200:ff:fe00:3	ICMPv6	90	Neighbor Advertisement
8. Reply	17 2.062452	2002:1::200:ff:fe00:5	2001:1::200:ff:fe00:1	ICMPv6	1090	Echo (ping) reply id=0
9. ARP	18 3.001856	fe80::200:ff:fe00:1	ff02::1:ff00:3	ICMPv6	90	Neighbor Solicitation
	19 3.006145	fe80::200:ff:fe00:3	fe80::200:ff:fe00:1	ICMPv6	90	Neighbor Advertisement
	20 3.006145	2001:1::200:ff:fe00:1	2002:1::200:ff:fe00:5	ICMPv6	1090	Echo (ping) request id=
	21 3.021123	2002:1::200:ff:fe00:5	2001:1::200:ff:fe00:1	ICMPv6	1090	Echo (ping) reply id=0

Obr. 2.50: Obsah souboru icmpv6-0-0.pcap - Shrnutí komunikace za pomoci informačních zpráv protokolu ICMPv6

1. Jako první probíhá statická konfigurace adres pro všechny uzly. Pokud si chce uzel adresu přidělit, musí zbylým uzlům oznámit jakou adresu hodlá používat. Tento proces se nazývá *DAD - Duplicate Address Detection* a využívá se k němu zpráva *NS - Neighbor Solicitation*, která je zasílána z nespecifikované adresy na multicastovou adresu skupiny, která by měla obsahovat všechny uzly v dané síti `ff02::1`. Pokud by se v síti nacházel uzel, který již vyhlašovanou adresu má, reagoval by na tuto zprávu zprávou *NA - Neighbor Advertisement*.
2. Uzel `host0` zasílá všem směrovačům (na multicastovou adresu všech směrovačů `ff02::2` zprávu *RS - Router Solicitation*. Pokud by uzel `host0` neměl

globální adresu nakonfigurovanou staticky, žádal by ní směrovače o zprávu *RA - Router Advertisment*, ve které by se dozvěděl, například zda se v síti nachází DHCPv6 server, pomocí kterého by si mohl adresu nakonfigurovat, nebo by mu v odpovědi směrovač zaslal prefixy, které jsou v této síti dostupné a podle kterých by si dokázal globální adresu nakonfigurovat sám pomocí bezstavové konfigurace SLAAC. Tuto zprávu uzel `host0` v tomto případě zasílá, kvůli získání dodatečných informací od směrovače jako například k zjištění adresy DNS serveru.

3. Předtím než bude moci uzel `host0` zaslat dotaz na dostupnost uzlu `host1` (bod 4. Echo Request), potřebuje do paketu doplnit správnou MAC adresu, kam má být tento paket zaslán. Uzel `host0` má nastavenou výchozí statickou cestu na směrovač `router0`, proto do paketu *Echo Request* potřebuje doplnit MAC adresu směrovače `router0`. Tu uzel `host0` zjišťuje pomocí funkcionality protokolu ICMPv6 zprávami NS a NA, které nahrazují protokol ARP pro IPv4. Zpráva NS je zasílaná na multicastovou *solicited-node* adresu uzlu `router0`. Tuto adresu si je každý uzel povinen vytvořit pro každou svou unicastovou adresu (globální, link-local) na každém ze svých rozhraní. Solicited-node adresa se vytváří za pomoci prefixu `ff02::1/104` a posledních 24b dané unicastového adresy. Uzel `host0` takto získá MAC adresu uzlu `router0` v odpovědi na zprávu NS, kterou je zpráva *NA - Neighbor Advertisement*.
4. Uzel `host0` tak už má všechny informace, které potřebuje pro zaslání zprávy *Echo Request*. Síťová adresa cíle je adresou uzlu `host1`, cílovou MAC adresou je MAC adresa směrovače `router0`, protože on je výchozí cestou pro uzel `host0`.
5. Takto se zpráva *Echo Request* dostala na uzel `router0`. Ten ji potřebuje dále přeposlat na směrovač `router1`. Avšak směrovač `router0` vidí, že pokud by uzel `host0` rovnou zaslal tuto zprávu na směrovač `router1`, se kterým je uzel `host0` ve stejné síti, byla by tato cesta efektivnější. Proto zasílá uzlu `host0` zprávu *Redirect*, aby si uzel přidal záznam do své směrovací tabulky o tom, aby provoz na uzel `host1` posílal na směrovač `router1`. Pro poslání zprávy potřebuje MAC adresu uzlu `host0`, tedy posílá zprávu NS ze žádostí o jeho MAC adresu.
6. Po získání MAC adresy uzlu `host0`, zasílá směrovač `router0` zprávu *Redirect*, aby si uzel `host0` upravil svou směrovací tabulku.
7. Dalšími zprávami jsou opět zprávy *NS* a *NA* pro získání MAC adres uzlů. Odpověď na první *NS* zprávu není v tomto souboru, jelikož přišla uzlu `router1`. A `router0` pomoci ní zjišťoval MAC adresu směrovače `router1`, aby mohl dále přeposlat zprávu Echo Request pro uzel `host1`. Další *NS* zprávou zjišťuje směrovač `router1` MAC adresu uzlu `host0`, aby mu uměl přeposlat odpověď Echo Reply.

8. Uzlu přichází odpověď na zprávu *Echo Request - Echo Reply*, přičemž IP adresou zdroje je adresa uzlu `host1`, MAC adresou zdroje je MAC adresa směrovače `router1`.
9. Na základě zprávy *Redirect* si uzel `host0` upravil svou směrovací tabulku a další *Echo Request* zprávy bude zasílat rovnou směrovači `router1`. Proto potřebuje získat jeho MAC adresou dalším dotazem pomocí dvojice zpráv *NS/NA*.

Protokol RIPng - RIP new generation

Udržování statických směrovacích informací není při měnících se sítích dlouhodobě efektivní. Proto i pro IPv6, tak jako pro protokol IPv4, existují různé dynamické směrovací protokoly. Jedním z nich je například obdoba protokolu RIP (Routing Internet Protocol) - protokol *RIPng (Routing Internet Protocol new generation)*. Pro definování směrování pomocí protokolu RIPng vymažte, případně zakomentujte, v kódu definování původního `InternetStackHelperu`. Jeho konfigurace bude mírně pozměněna, kvůli spolupráci s protokolem RIPng.

```
//InternetStackHelper internetv6;dsfds
//internetv6.SetIpv4StackInstall (false);fdsfsd
//internetv6.Install (all); fdsfsd
```

Dále vymažte nebo zakomentujte přidání statických cest, které již nebudou potřebné, o směrování se postará protokol RIPng.

```
//interfaces_net1.SetDefaultRoute (0,1);
//interfaces_net1.SetDefaultRoute (1, 2);
//interfaces_net2.SetDefaultRouteInAllNodes (0);
```

Za příkaz `NetDeviceContainer netDevices_net2 = csma.Install (net2);` zkopírujte nasledujúci kód, který definuje RIPng směrování pro všechny uzly v topologii.

```
RipNgHelper ripNgRouting;
Ipv6ListRoutingHelper ripRoutingHelper;
ripRoutingHelper.Add (ripNgRouting, 0);

InternetStackHelper internetv6;
internetv6.SetIpv4StackInstall (false);
internetv6.SetRoutingHelper (ripRoutingHelper);
internetv6.Install (all);
```

Před spuštěním simulace ještě upravte čas prvního výpisu směrovací tabulky uzlu `host0` na čas 0s, druhý výpis můžete ponechat na čas 3s.

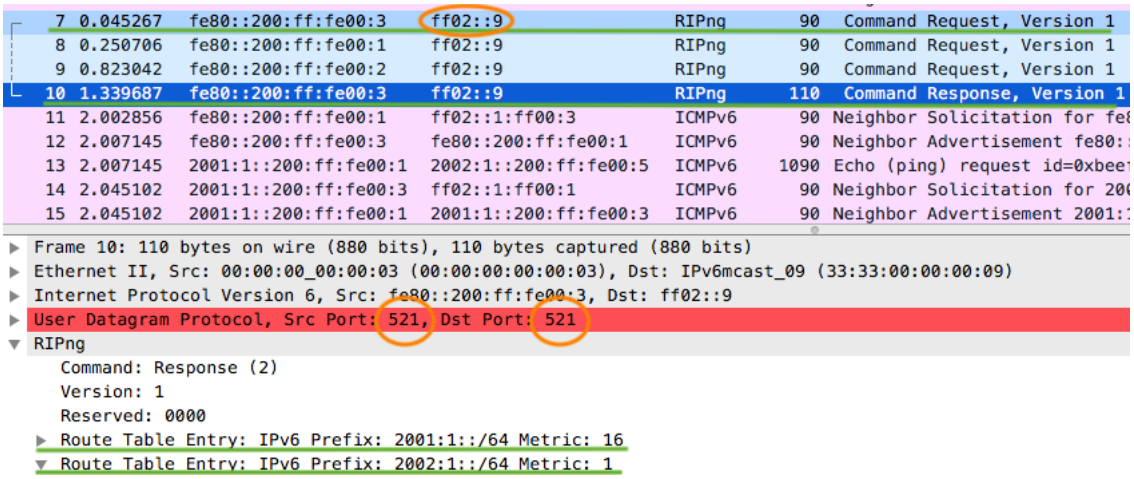
```
routingHelper.PrintRoutingTableAt (Seconds (0.0),
    host0, routingStream );
```

Po spuštění simulace se podívejte na konzolový výstup se směrovacími tabulkami. Tento výstup je také zobrazen na obrázku 2.51. V směrovacích tabulkách je vidět,

```
Node: 0, Time: +0.0s, Local time: +0.0s, Ipv6ListRouting table
  Priority: 0 Protocol: ns3::RipNg
Node: 0, Time: +0.0s, Local time: +0.0s, IPv6 RIPng table
Destination                    Next Hop                   Flag Met Ref Use If
2001:1::/64                    ::                         U    1   -   -   1

Node: 0, Time: +3.0s, Local time: +3.0s, Ipv6ListRouting table
  Priority: 0 Protocol: ns3::RipNg
Node: 0, Time: +3.0s, Local time: +3.0s, IPv6 RIPng table
Destination                    Next Hop                   Flag Met Ref Use If
2002:1::/64                    fe80::200:ff:fe00:3        UG   2   -   -   1
2001:1::/64                    ::                         U    1   -   -   1
```

Obr. 2.51: Výpis směrovací tabulky uzlu host0 v čase 0s a 3s podle protokolu RIPng

že uzel `host0` si již nepřidal do směrovací tabulky záznam pouze pro uzel `host1`, ale záznam o směrovaní provozu pro celou síť 2 (`2002:1::/64`). Tuto informaci nezískal hned, protože v době 0s ještě záznam pro síť 2 nemá, dostal ho až ve zprávě od protokolu RIPng. Otevřete si soubor s pakety uzlu `host0 icmpv6-0-0.pcap`. Jeho obsah by měl shodovat s obrázkem 2.52. Při zobrazení detailu paketu protokolu



Obr. 2.52: Obsah souboru icmpv6-0-0.pcap - Detail zprávy protokolu RIPng

RIPng je vidět, že protokol používá číslo portu 521. Jako první je zaslána protokolem RIPng zpráva typu *Command Request*, v níž směrovač žádá ostatní směrovače o jejich směrovací tabulky. Zpráva je zasílána na multicastovou adresu skupiny všech RIPng uzlů - `ff02::9`. Obdobné multicastové adresy má IPv6 definováno i například pro směrovače, které používají směrovací protokol OSPFv3 - `ff02::5`, aby si směrovače mohly posílat směrovací informace jenom mezi sebou a nezatěžovaly nimi

ostatní uzly. Odpovědí na RIPng zprávu *Command Request* je typ *Command Response*, ve které uzel uvádí sítě ze své směrovací tabulky. Protokol se pro rozhodování o výběru cesty řídí hlavně na hodnotou metriky pro danou síť. Metrika RIPng sítě je celé číslo od 1 do 15 včetně. Je možné se setkat i s hodnotou 16, která značí nekonečno. Tuto hodnotu je vidět i v odpovědi směrovače `router1`. Ta je tam z důvodu získání informace o síti od statického `routingHelperu` simulátoru NS, který měl pro síť definovanou hodnotu 0 a protože protokol RIPng nemá hodnotu 0 ve svém definovaném rozmezí, dal této síti hodnotu nekonečna - 16. Pro síť 2 již uvádí správnou metriku 1. Tyto údaje přijal uzel `host0`, který si na základě nich už zprávě aktualizoval svou směrovací tabulku - pro svou síť 1 mu zbývá metrika 1, ale pro síť 2 si k původní hodnotě 1 přičítá hodnotu z RIPng odpovědi - 1. Tedy ví, že k síti 2 potřebuje paket 2 hopy. Tato odpověď je také zaslána všem uzlům s protokolem RIPng, aby si i oni upravili své tabulky. Zprávy *Command Response* a *Command Response* si uzly přeposílají v přeposílají v pravidelných intervalech, čímž se uzly dozvědí i o změně sítě.

2.2.5 3. časť: Chybové zprávy

Zprávy *NS/NA*, *RS/RA*, *Echo Request/Reply* a *Redirect* spadají do kategorie informačních zpráv protokolu ICMPv6. Jak bylo zmíněno v úvodní kapitole 2.2.1, protokol ICMPv6 má také zprávy ohlašující chyby, které v síti mohly nastat. Těmito zprávami jsou například zprávy *Destination unreachable*, *Time Exceeded* či *Packet Too Big*.

Zprávy Time Exceeded a Destination Unreachable

Zpráva *Time Exceeded* bude vyvolána, pokud směrovač sníží hodnotu Hop limit v paketu na hodnotu 0 nebo příjme paket s hodnotou Hop limit rovnou 0. Takový paket směrovač dále zahazuje. Výchozí hodnota pro hop limit pro paket se zprávou *Echo Request* je stanovena na 64, pro vyvolaní chyby a zprávy *Time Exceeded* je potřeba upravit hop limit paketu, aby při pokusu o doručení klesl na hodnotu 0. Následující řádek nastaví hop limit na hodnotu 1. Řádek přidejte hned na začátek funkce `main()`. Při každém vkládání kódu dbejte aby kopírovaný příkaz neobsahoval mezery, ty mohou způsobovat chyby při překladu programu. String `"ns3::Ipv6L3Protocol::DefaultTtl"` nemůže obsahovat mezeru.

```
Config::SetDefault(
    "ns3::Ipv6L3Protocol::DefaultTtl", UintegerValue (1));
```

Zpráva *Destination unreachable* by měla být vygenerována směrovačem jako odpověď na paket, který nelze doručit na cílovou adresu z jiných důvodů než kvůli přetížení

sítě. To se například stane, pokud uzlu vypadne spojení se směrovačem. Tuto chybu lze v síti vytvořit pomocí následující funkci - `UnconnectNode()`. Přidejte ji ještě před funkci `main()`.

```
void UnconnectNode (Ptr<Node> node, uint32_t intNumber){
  node->GetObject<Ipv6>()->SetDown(intNumber); }
```

Dále vložte volání funkce `UnconnectNode()` před začátek simulace. Funkce v čase 5s zruší přepojení uzlu `host1` se směrovačem `router1`, což vyvolá zprávu *Destination unreachable*, jelikož zpráva *Echo Request* nebude moci být uzlu `host1` doručena.

```
Simulator::Schedule (
    Seconds (5), &UnconnectNode, router1, 2);
```

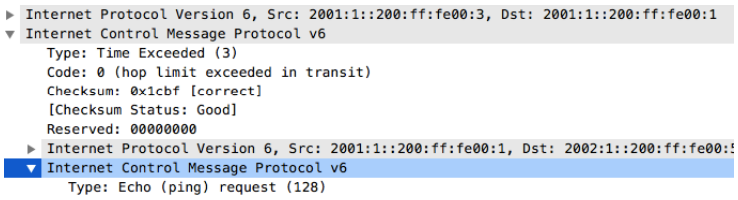
Spusťte znovu simulaci a otevřete si pcap soubor uzlu `host0` - *icmpv6-0-0.pcap*. Jeho obsah by měl shodovat s obrázkem 2.53.

No.	Time	Source	Destination	Protocol	Length	Info
15	2.020034	2001:1::200:ff:fe00:1	2001:1::200:ff:fe00:3	ICMPv6	90	Neighbor Advertisement 2
16	2.026000	2001:1::200:ff:fe00:3	2001:1::200:ff:fe00:1	ICMPv6	1138	Time Exceeded (hop limit
17	2.997856	2001:1::200:ff:fe00:1	2002:1::200:ff:fe00:5	ICMPv6	1090	Echo (ping) request id=0
18	3.005421	2001:1::200:ff:fe00:3	2001:1::200:ff:fe00:1	ICMPv6	1138	Time Exceeded (hop limit
19	3.108997	fe80::200:ff:fe00:1	ff02::9	RIPng	110	Command Response, Versi
20	3.997856	2001:1::200:ff:fe00:1	2002:1::200:ff:fe00:5	ICMPv6	1090	Echo (ping) request id=0
21	4.005421	2001:1::200:ff:fe00:3	2001:1::200:ff:fe00:1	ICMPv6	1138	Time Exceeded (hop limit
22	4.853608	fe80::200:ff:fe00:2	ff02::9	RIPng	110	Command Response, Versi
23	4.997856	2001:1::200:ff:fe00:1	2002:1::200:ff:fe00:5	ICMPv6	1090	Echo (ping) request id=0
24	5.005421	2001:1::200:ff:fe00:3	2001:1::200:ff:fe00:1	ICMPv6	1138	Destination Unreachable
25	5.997856	2001:1::200:ff:fe00:1	2002:1::200:ff:fe00:5	ICMPv6	1090	Echo (ping) request id=0
26	6.005421	2001:1::200:ff:fe00:3	2001:1::200:ff:fe00:1	ICMPv6	1138	Destination Unreachable
27	7.007144	fe80::200:ff:fe00:3	fe80::200:ff:fe00:1	ICMPv6	90	Neighbor Solicitation fo

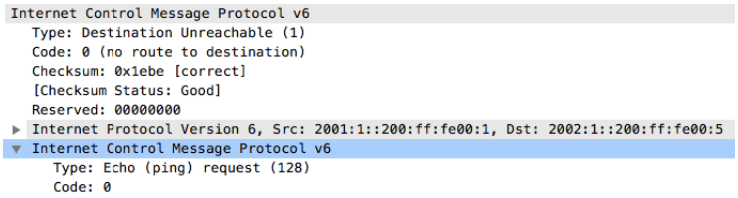
Obr. 2.53: Obsah souboru icmpv6-0-0.pcap - Chybové zprávy protokolu ICMPv6 Time Exceeded a Destination Unreachable

Obě chybové zprávy byly vytvořeny směrovačem `router1`, protože on snížil hodnotu Hop limit z 1 na 0. Hodnota Hop limit paketu je snižována každým směrovačem, kterým projde. Pro tento paket by to znamenalo, že prošel 63 směrovači, což je poměrně hodně a může to znamenat například, že v síti se nachází směrovací smyčka. Přesná struktura zprávy je na obrázku 2.54. Zpráva *Time Exceeded*, spolu se zprávou *Destination unreachable*, která je zobrazena na obrázku 2.55, mají podobný formát, jehož hlavními poli jsou:

- *Typ* - tak jako pro informační zprávy, toto pole udává typ dané ICMPv6 zprávy. Pro zprávu *Time Exceeded* je to číslo 3 a pro *Destination unreachable* 1.
- *Kód* - toto pole má také za úkol detailnější popsat důvod proč byla tato zpráva vyvolána. Narozdíl od informačních zpráv, mají chybové zprávy pro kód více definovaných hodnot, které přibližují důvod vzniku dané chyby.
  Pro zprávu *Time Exceeded* je to hodnota 0, která říká, že paketu během přenosu v síti klesla hodnota hop limit na 0. A může obsahovat ještě hodnotu 1,



Obr. 2.54: Detail zprávy Time Exceeded



Obr. 2.55: Detail zprávy Destination Unreachable

kdy vypršel čas rekonstrukce fragmentu.

Pro zprávu *Destination unreachable* je také definována hodnota 0, která definuje, že neexistuje cesta k danému cíli. Pro tento typ zprávy je definovaných ještě dalších 6 kódů jako například - Address / Port unreachable, komunikace byla administrativně zakázána apod.

Jak již bylo zmíněno, zpráva *Time Exceeded* souvisí s nulovou hodnotou Hop limit a zpráva *Destination unreachable* byla vyvolána po odpojení uzlu `host1` od směrovače `router1`. O tomto výpadku a nemožnosti doručení zprávy *Echo Request* směrovač `router1` takto informoval uzel `host0`, že zpráva uzlu nemohla být doručena.

Zpráva Packet Too Big

Poslední představenou zprávou je *Packet Too Big*. Tato zpráva musí být zaslána směrovačem jako odpověď, když směrovač nemůže paket zaslat dále kvůli jeho velikosti, která je větší než MTU (Maximum Transmission Unit) pro dané odchozí spojení. To se v tomto případě stane, pokud velikost zasílaného paketu bude například 4096B a MTU v síti 1 bude mít standardní hodnotu 1500B, ale síť 2 bude mít MTU o velikosti 5000B. Přidejte do kódu změnu MTU pro síť 2 v definici spojů mezi řádky `NetDeviceContainer netDevices_net1 = csma.Install (net1);` a `NetDeviceContainer netDevices_net2 = csma.Install (net2);` Při každém vkládání kódu dbejte aby kopírovaný příkaz neobsahoval mezery, ty mohou způsobovat chyby při překladu programu. Stringy `"Mtu"`, `"PacketSize"` nemohou obsahovat mezeru.

```
csma.SetDeviceAttribute ("Mtu", UintegerValue (5000));
```

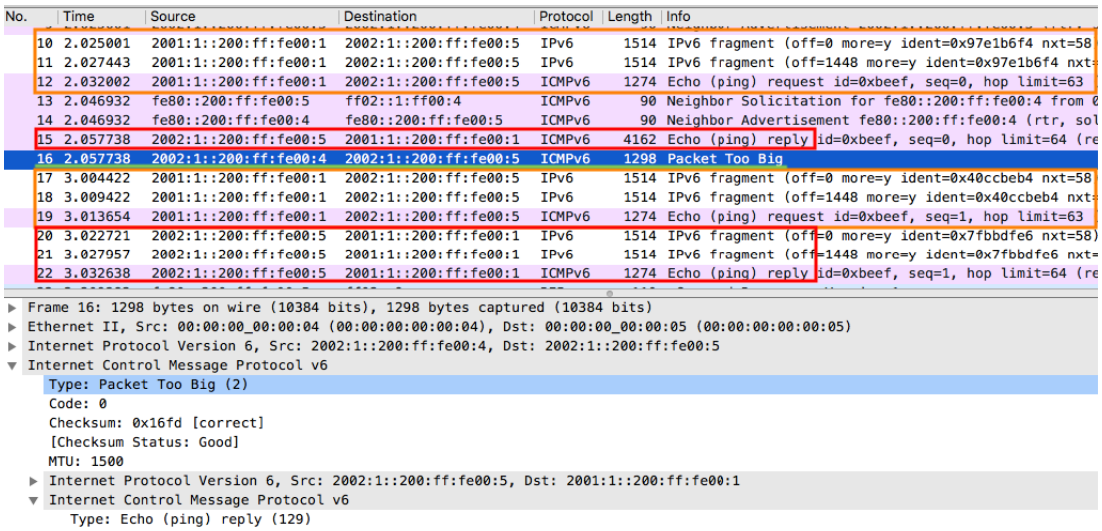
V definici aplikace Ping změňte velikost paketu na hodnotu 4096B.

```
ping6.SetAttribute ( "PacketSize" , UintegerValue (4096));
```

Před spuštěním simulace ještě nezapomeňte zakomentovať snížení hodnoty Hop limit a vyvolání odpojení uzlu `host1` od směrovače `router1`.

```
//Config::SetDefault(
    "ns3::Ipv6L3Protocol::DefaultTtl", UintegerValue (1));
//Simulator::Schedule (
    Seconds (5), &UnconnectNode, router1, 2);
```

Po dokončení simulace si otevřete soubor *icmpv6-2-1.pcap*, ve kterém jsou odeslané a přijaté pakety směrovače `router1` na jeho rozhraní s uzlem `host1`. Viz, co se stalo s paketem, který přenášel zprávu *Echo Request*, kdy byla vygenerenovaná zpráva *Packet Too Big* a jak byla odeslána zpráva *Echo Reply*. Obsah souboru by se měl shodovat s obrázkem 2.56.



Obr. 2.56: Obsah souboru icmpv6-2-1.pcap - Detail chybové zprávy Packet Too Big

Zpráva *Echo Request* byla odeslána ze sítě 1, kde uzel `host0` musel sám paket o velikosti 4096B před odesláním nejdříve rozdělit na jednotlivé fragmenty, kvůli MTU 1500B. Pro odpověď *Echo Reply*, má uzel `host1` v síti 2 už MTU 5000B, čili dokáže celý paket zaslat najednou. Avšak při příchodu paketu *Echo Reply* na směrovač `router1`, směrovač vidí, že MTU pro sít 1 je jen 1500B. Pokud by tato situace nastala v rámci protokolu IPv4, směrovač by byl tuto zprávu rozdělil na fragmenty sám. To je však pro něj další zátěž způsobující zpoždění, přičemž toto rozdělení může udělat i samotný zdrojový uzel. Řešením tohoto problému je v rámci IPv6 zpráva *Packet Too Big*. Tu zasílá směrovač `router1` uzlu `host1` a dává mu v ní informaci o tom, že pro další směrování paketu je třeba odpověď *Echo Reply* rozdělit

na fragmenty, které budou vyhovovat hodnotě MTU dalšího skoku. Hlavními poli zprávy jsou:

- *Typ* - pole číselného označení typu zprávy, pro tuto zprávu je to číslo 2.
- *MTU* - pole, ve kterém je uvedena hodnota MTU, kterou má další skok a podle níž má být paket rozdělen na fragmenty.

Jak je vidět v paketech č. 20 - 22, další odpověď *Echo Reply* už uzel `host1` rozdělil na fragmenty podle MTU uvedeného ve zprávě *Packet Too Big*, by ji směrovač `router1` mohl zaslat do sítě 1.

2.2.6 Kontrolní otázky

1. Jaké je základní dělení zpráv protokolu ICMPv6?
2. Jaký druh IPv6 adresy může rozhraní vygenerovat hned po jeho startu aby uzel mohl komunikovat v rámci lokální sítě? Jaký prefix má tato adresa?
3. Jak se při přidělování IPv6 adres zajistí, aby nebyla stejná adresa použita vícekrát?
4. Kdy bude v síti vygenerována zpráva Redirect?
5. Jakými zprávami nahrazuje protokol ICMPv6 funkcionalitu protokolu ARP v rámci IPv4? Jakou výhodu má solicited-node adresa oproti ARP broadcastu?
6. Jaký je rozdíl mezi zprávami Time Exceeded a Destination unreachable?
7. Kde dochází ke fragmentaci IPv6 paketů?

Záver

Cieľom tejto diplomovej práce bolo navrhnúť a vytvoriť dve laboratórne úlohy v simulačnom prostredí ns-3 pre vysvetlenie princípov vybraných komunikačných protokolov. Prvým z vybraných protokolov bol protokol BGP, ktorý slúži k riadeniu smerovania na úrovni autonómnych systémov. Ako druhý bol vybraný protokol ICMPv6, ktorý je základným protokolom sady IPv6 a slúži k ohlasovaniu chýb v sieti, testovaniu dosiahnuteľnosti a k výmene niektorých informácii o prevádzke v sieti. Tieto protokoly sú dôležitou súčasťou prenosu dát v rámci Internetu, ich znalosť je dôležitá pre správnu konfiguráciu zariadení, ktoré tento prenos zabezpečujú koncovým zariadeniam a používateľom.

Teoretická časť práce bola zameraná na opis správ pomocou ktorých vybrané protokoly komunikujú a na zistenie spôsobu preposielania týchto správ, za účelom získania požadovanej funkcionality protokolu.

Pre protokol BGP to boli správy Open, Update, Notification a Keepalive spolu s opisom spôsobu, akým si smerovače medzi sebou vytvárajú susedstvá a ako si v rámci nich vymieňajú potrebné informácie. Pre túto úlohu boli ďalej v teoretickej časti práce preskúmané tri možnosti použitia protokolu BGP v prostredí simulátoru ns-3. Dve z týchto možností boli vytvorené v rámci súkromných projektov používateľov prostredia ns-3 a sú voľne dostupné k použitiu. Ani jedna z týchto dvoch možností však neobsahovala úplnú podporu použitia protokolu BGP. Najvhodnejšou voľbou, ktorá bola využitá aj pre tvorbu laboratórnej úlohy, bola možnosť použitia aplikácie Quagga spolu s modulom priameho vykonávania kódu pre prostredie ns-3. Za pomoci aplikácie Quagga bolo možné vytvoriť topológiu s viac ako troma autonómnymi systémami, ktoré si medzi sebou dokázali správne vytvoriť susedstvá a simulovať tak komunikáciu na úrovni hraničných smerovačov autonómnych systémov.

Pre protokol ICMPv6 boli v tejto kapitole uvedené informačné správy Neighbor Solicitation, Neighbor Advertisement, Router Solicitaion, Router Advertisement, Redirect, Echo Request a Echo Reply, a chybové správy Destination Unreachable, Time Exceeded a Packet Too Big. V rámci opisu funkcionality informačných správ protokolu boli v kapitole uvedené dva spôsoby konfigurácie IPv6 adries, mechanizmus vytvárania IPv6 adresy z linkovej adresy, mechanizmus zisťovania duplikátnych adries, spôsob získania linkovej adresy zo sieťovej adresy uzlu a spôsob zisťovania dostupnosti uzlu. V rámci opisu chybových správ bola pre každú zo správ uvedená situácia, aká musí nastať, aby bola daná správa vytvorená a danú chybu oznámila.

Praktickou časťou práce bol návrh a vytvorenie samotných laboratórnych úloh, v ktorých sú využité a opísané jednotlivé funkcionality protokolov spolu so správami, ktoré k danej funkcionalite využívajú. Na začiatku oboch úloh má študent k dispo-

zícii vopred vytvorenú topológiu, s ktorou pracuje a pre ktorú je simulácia spúšťaná. Po spustení simulácie študent pracuje s výstupmi dát, ktoré danú simuláciu opisujú. Výstupmi sú súbory typu pcap, ktoré obsahujú pakety uzlov, výpisy smerovacích tabuliek uzlov či vizualizácia toku dát pre protokol BGP a výpis jednotlivých adries pridelených rozhraniam v rámci protokolu ICMPv6.

V úlohe pre protokol BGP je vysvetlený a ukázaný spôsob vytvárania susedstiev smerovačov jednotlivých autonómnych systémov, reakcia protokolu BGP na výpadok v sieti a vytvorenenie tzv. PeerLinku v rámci susedstva smerovačov. V samostatnej časti má študent za úlohu do vzniknutej topológie pridať ďalší autonómny systém a overiť si jeho správnu konfiguráciu.

Úloha pre protokol ICMPv6 sa skladá z troch častí. Prvá časť sa zaoberá statickou a bezstavovou konfiguráciou adries IPv6 a mechanizmami, ktoré s nimi súvisia. Táto časť obsahuje tri samostatné podúlohy, v ktorých sa má študent bližšie zoznámiť s priebehom mechanizmov DAD a EUI-64 a bezstavovej konfigurácie. V ďalšej časti je do úlohy pridané posielanie správ o dostupnosti uzlov a dve možnosti nastavenia smerovania v rámci IPv6 sietí - statické smerovanie za pomoci nastavenia východzích ciest a dynamické za pomoci protokolu RIPng. Posledná časť úlohy je venovaná vyvolaniu troch chýb, ktoré v IPv6 sieťach môžu nastať a spôsobu, akým na ne protokol ICMPv6 dokáže reagovať. Každá z častí úlohy obsahuje ukážku a opis paketov s príslušnými správami protokolu ICMPv6, ktoré s danými funkcionalitami súvisia.

Vytvorené úlohy by mali študentom ukázať a objasniť spôsoby, akými protokoly v sieťach pracujú a praktický dopad jednotlivých teoretických vlastností vybraných protokolov.

Literatúra

[1] JEŘÁBEK J. Pokročilé komunikační techniky [skriptum], 2020.

[2] GOBRIAL M.N. Evaluation of border gateway protocol (bgp) version 4 (v4) in the tactical environment. In *Proceedings of MILCOM '96 IEEE Military Communications Conference*, volume 2, pages 490–495 vol.2, 1996. doi:10.1109/MILCOM.1996.569372.

[3] GROSS P. G., REKHTER Y. Application of the Border Gateway Protocol in the Internet. RFC 1772, March 1995. URL: https://rfc-editor.org/rfc/rfc1772.txt, doi:10.17487/RFC1772.

[4] REKHTER Y., HARES S. a LI T. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006. URL: https://rfc-editor.org/rfc/rfc4271.txt, doi:10.17487/RFC4271.

[5] IPCISCO.COM. Upravený obrázok výmeny bgp správ. [online]. URL: https://ipcisco.com/lesson/bgp-peers-bgp-sessions-bgp-messages/.

[6] KOLEKTÍV VÝVOJÁROV ns3. *ns-3 Software Architecture*, October 2007. URL: https://www.nsnam.org/docs/architecture.pdf.

[7] SAHRAEI; M. R. Computer networking integrating bgp with ns-3.13 network simulator (ns3-bgp). Technical report, Simon Fraser University, 2012. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.278.5926&rep=rep1&type=pdf.

[8] MORICHIKA N. libbgp - bgp (border gateway protocol) library. Technical report, 2019. URL: https://lab.nat.moe/ns3bgp-doc/.

[9] KOLEKTÍV VÝVOJÁROV Quagga. *Quagga manual Direct Code Execution project*, Marec 2015. URL: https://www.nsnam.org/docs/dce/manual-quagga/html/getting-started.html#building-ns-3-dce-and-dce-quagga.

[10] MORICHIKA N. ns-bgp bgp module for ns-3. Technical report, 2019. URL: https://github.com/Nat-Lab/libbgp.

[11] TAZAKI , HAJIME , UARBANI, a i. Direct code execution: Revisiting library os architecture for reproducible network experiments. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, page 217–228, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2535372.2535374.

[12] SATRAPA P. *IPv6 : internetový protokol verze 6*. CZ.NIC, z.s.p.o, Praha, 2019. URL: https://www.bookport.cz/kniha/ipv6-ctvrte-vydani-5998/.

[13] GUPTA M., CONTA A. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443, March 2006. URL: https://rfc-editor.org/rfc/rfc4443.txt, doi:10.17487/RFC4443.

[14] GAI, SILVANO, CAPORELLO. *Internetworking IPv6 with Cisco routers*. McGraw-Hill, 1998. URL: http://www.cu.ipv6tf.org/literatura/chap5.pdf.

[15] KOZIEROK Ch. M. *The TCP/IP-Guide: a comprehensive, illustrated Internet protocols reference*. No Starch Press, 2009. URL: http://www.tcpipguide.com/free/t_InternetControlMessageProtocolICMPICMPv4andICMPv6.htm.

[16] SIMPSON W. A., NARTEN T. Dr., NORDMARK E. , SOLIMAN H. Neighbor Discovery for IP version 6 (IPv6). RFC 4861, September 2007. URL: https://rfc-editor.org/rfc/rfc4861.txt, doi:10.17487/RFC4861.

[17] NARTEN T. Dr., JINMEI T., THOMSON S. Dr. IPv6 Stateless Address Autoconfiguration. RFC 4862, September 2007. URL: https://rfc-editor.org/rfc/rfc4862.txt, doi:10.17487/RFC4862.

[18] IBM Knowledge Center. Ipv6 protocol: Duplicate address detection, 2014. URL: https://www.ibm.com/docs/en/zos/2.1.0?topic=discovery-duplicate-address-detection.

[19] BOVY F. Ipv6 autoconfig, 2011. URL: https://www.slideshare.net/fredbovy/ipv6-autoconfig.

[20] CISCO PRESS. Mastering ipv6 slaac concepts and configuration. *www.ciscopress.com*, 2013. URL: https://www.ciscopress.com/articles/article.asp?p=2154680.

[21] KOLEKTÍV VÝVOJÁROV ns3. *ns-3 IPv6 - Model Library*. URL: https://www.nsnam.org/docs/release/3.30/models/html/ipv6.html#auto-generated-ipv6-addresses.

[22] stretch (PSEUDONYM). Eui-64 in ipv6, 2008. URL: https://packetlife.net/blog/2008/aug/04/eui-64-ipv6/.

[23] FISHBURNE D. Understanding ipv6: The ping before solicited-node multicast. 2014. URL: https://www.networkcomputing.com/networking/understanding-ipv6-ping-solicited-node-multicast.

[24] FISHBURNE D. Understanding ipv6: Solicited-node multicast in action. 2014. URL: https://www.networkcomputing.com/networking/understanding-ipv6-solicited-node-multicast-action.

[25] FISHBURNE D. Understanding ipv6: What is solicited-node multicast? 2014. URL: https://www.networkcomputing.com/networking/understanding-ipv6-what-solicited-node-multicast.

[26] JUNIPER NETWORKS-Product and Release Support. Rip and ripng overview, 2021. URL: https://www.juniper.net/documentation/us/en/software/junos/rip/topics/topic-map/rip-and-ripng-overview.html.

[27] VENAAS S. Ipv6 multicast address space registry. Technical report, IANA, Marec 2021. URL: https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml.

[28] DEERING S. E. Dr., HINDEN B. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, July 2017. URL: https://rfc-editor.org/rfc/rfc8200.txt, doi:10.17487/RFC8200.

[29] MCCANN J., DEERING S. E., MOGUL J., HINDEN B. Path MTU Discovery for IP version 6. RFC 8201, July 2017. URL: https://rfc-editor.org/rfc/rfc8201.txt, doi:10.17487/RFC8201.

Zoznam symbolov a skratiek

AS	Autonómny systém
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
DAD	Duplticate Address Detection
DCE	Priame vykonávanie kódu – Direct Code Execution
EGP	Exterior Gateway Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
EUI-64	Extended Unique Identifier
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol verzie 6
IGP	Interior Gateway Protocol
IS-IS	Intermediate System-to-Intermediate System
NA	Neighbor Advertisement
ND, NDP	Neighbor Discovery, Neighbor Discovery Protocol
NLRI	Network Layer Reachability Information
NS	Neighbor Solicitation
OSPF	Open Shortest Path First Protocol
RA	Router Advertisement
RIP	Routing Information Protocol
RIPng	Routing Information Protocol next generation
RS	Router Solicitation
TCP	Transmission Control Protocol